RADIK: AN INTERACTIVE GRAPHIC AND TEXT EDITOR SYSTEM

Ralph Henry Stowell

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

RADIK: AN INTERACTIVE GRAPHIC
AND TEXT EDITOR SYSTEM

by

Ralph Henry Stowell, Jr.

Thesis Advisor: George A. Rahe

June 1972

Approved for public release; distribution unlimited.

RADIK: An Interactive Graphic and Text Editor System

by

Ralph Henry Stowell, Jr.
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1972

ABSTRACT

This thesis contains a survey of the hardware requirements and the desirable characteristics of an interactive Computer Graphics System. There is a detailed evaluation of the resident system in the Computer Laboratory and an attempt is made through programming, to provide a software capability which allows for full utilization of all hardware in the Computer Graphics System. This program, known as RADIK, is a highly interactive graphics and text editor developed for utilization on a computer system composed of an XDS 9300 and two Adage AGT-10 graphics terminals. In addition, this thesis includes a discussion on the development of the system as well as complete operating instructions.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The author wishes to express his appreciation to William Thomas of the Computer Laboratory, Naval Postgraduate School, for his time and patience in assisting in the understanding of the program GATED and the XDS 9300 assembly language.

I. INTRODUCTION

Computer graphics is one of the most rapidly expanding developments in the field of electronic data processing today. The primary reason for this is because, unlike most other computer operations it allows for a high degree of interaction between man and machine. The computer provides the high speed manipulation of large amounts of data, while the man contributes experience, judgment and imagination. Additionally, as Licklider [1] accurately stated, "perception is more rapid when a concept can be visualized rather than pre-processing alpha-numeric strings before we can think about the real concepts."

A. PURPOSE

In the design of an interactive system there are many considerations to be taken into account and trade offs to be made. Some of these will be discussed in this paper along with the evaluation of a particular system and recommended changes for improving it. To this end the purpose of this thesis is threefold.

1. To examine the elements and desirable characteristics of any interactive graphics system.
2. To describe and evaluate the resident graphics system in the Computer Laboratory at the Naval Postgraduate School.
3. Finally, to discuss RADIK, a graphics and text editor program, developed in association with Lieutenant R.F. Ashford, Jr., which allows for greater interaction with the present system, therefore, increasing its capabilities.

B. ORGANIZATION

The thesis is divided into 7 parts, exclusive of apendices and computer program.

Section II is a brief discussion of the historical background of the "state of the art" in computer graphics.

Section III contains a discussion and analysis of the various hardware characteristics which must be considered in the design of an interactive computer graphics system.

Section IV is an investigation into the overall, graphical and interactive characteristics which are desirable in any Computer Graphics System.

In Section V there is a description of the computer graphics system installed in the Computer Laboratory. This description includes a brief coverage of the hardware and a detailed description of GATED, the graphics and text editor program in use at present.

Section VI deals with RADIK, a highly interactive graphics and text editor program, which has been developed in order to replace GATED. In addition, there is a discussion of the objectives of RADIK as well as the programming modifications which were made. This section also includes a detailed explanation of the procedures required in the operation of the program.

Section VII is devoted to conclusions.

II. HISTORICAL BACKGROUND

A. DEFINITION OF COMPUTER GRAPHICS

Prior to a description of the early graphics systems, there should be an understanding of the term Computer Graphics. The Merriam Webster New Collegiate Dictionary (7th ed.) defines graphics as "the art or science of drawing a representation of an object upon a two dimensional surface according to mathematical rules of projection." Technically then, we could extend this definition to "Computer Graphics" by simply adding the phrase, for display by a computer. While this would in fact be a correct definition it does not convey the true meaning in which Computer Graphics are thought of today. In the past few years technological developments in computer systems and information handling have been directed toward an improved interaction between man and machine. This is witnessed by the fact that a great deal of attention has been given to improving the languages in which computer instructions are written. It may also be evidenced by the fact that a major part of the research in Computer Graphics has been conducted in the area of graphics languages and interactive systems. This has been done to provide better "real time" communications between the computer and the operator. Let us then redefine the meaning of Computer Graphics as does Mr. C.I. Johnson in his article "Principles of Interactive Systems" [2]. He states, "Computer Graphics has become an accepted term to denote that set of computer techniques and applications wherein data is either presented or accepted by a computer in the form of line drawings or graphics."

B. PIONEER SYSTEMS

Having defined the term Computer Graphics a discussion of the early systems follows.

Perhaps the one most significant advancement in hardware design to affect the development of Computer Graphics was the time-sharing and on-line system. With the advent of these two pieces of equipment it wasn't long before pioneers in the Computer Graphics field were developing a real time visual display with which individual userscould interact.

1. Whirlwind

One of the first of these systems to be developed was at MIT in the early 1950's. Mr. J. Forrester and a group of colleagues, known as the Whirlwind Group [1] developed a graphical display of an air defense situation which allowed for user interaction by use of a light gun. This light gun was the forerunner of light pens used in today's graphics systems and the Whirlwind program was later to become an integral part of SAGE [3], an air defense system used in the United States and Canada. While the Whirlwind Computer allowed for a very limited amount of interaction, it was at least a start at man machine symbiosis [4].

2. Sketchpad

Perhaps the next most significant development was Sutherland's SKETCHPAD program [5] in which the use of a CRT display, and a light pen to draw pictures, was demonstrated. In this program the computer monitored the operator's motions and built a data structure which was used to represent the properties of the drawing. In addition,relations between elements of the system in the drawing could be specified using the light pen, by first selecting the element and then the relation.

The major implications of the SKETCHPAD program were that it not only allowed the input of data at the CRT, but also made possible the control of the program by an external device. This particular program was later extended to three dimensions by C. Johnson in SKETCHPAD III [6].

3. DAC-1

In the mid 1960's DAC-1 [7] (Design Augmented by Computer), a prototype of the IBM 2250 Graphics System [8] was developed by General Motors. The DAC-1 was an experimental system in which a designer, seated at a console, could create or modify the designs of parts of automobiles. In addition, these sketches would then become data in the computer's memory and with the aid of supporting programs would produce wooden models of the sketched parts.

III. HARDWARE REQUIREMENTS OF AN INTERACTIVE GRAPHICS SYSTEM

In any interactive graphics system there are certain components which are basic to the operation, while others are considered desirable. The system must contain at a bare minimum; a digital computer, some means of displaying the desired graphical output, and finally, an interface through which the user is able to communicate with the computer; a light pen, keyboard or similar device.

A. COMPUTER CONFIGURATION

One of the most important items in any operation, is the heart of the system, the computer itself. In Computer Graphics there are two basic configurations which are utilized. The first and most popular, is the combination of a small digital computer linked by means of a high speed communications system to a large multiprocessing unit. In this type of operation the small computer ensures the completion of all graphical housekeeping chores, while the host computer has the responsibility for programming operations and associated activities. The second, generally less desirable, configuration is to utilize one medium size digital computer which is dedicated strictly to the graphical problem.

The relative merits of these two systems will be discussed in a later section.

B. CATHODE RAY TUBE

1. Description

The most widely used display device and the only one discussed in this paper is the Cathode Ray Tube. The tube itself consists of a phosphorus coated surface on which an illuminated trace is generated by

means of an electron gun. The gun fires the electrons at the phosphorus covered tube and x/y positioning is obtained electronically by deflecting the electrons utilizing magnetic coils.

2. Visual Parameters to be Considered

When using the CRT for display purposes there are two parameters which should be considered critical to the operation of the tube. Those are the decay rate of the phosphor and the refresh rate of the image.

The decay rate of the phosphor must be fast enough that a residual image of an old trace does not become confused with a new trace. In order to ensure the above requirement is met and yet keep the image on the screen, the refresh rate should be fast enough that a flicker [9] free picture is obtained. Normally a refresh rate of 40 times per second is sufficient to keep the probability of visual annoyance at a low level [10].

This presents a case for the use of a "time-shared" system which has a large computer capable of handling multiprocessing tasks. While the small graphics package is taking care of the refreshing of the picture, the large parent computer may proceed with the program being executed and process user generated interrupts.

C. COMMUNICATIONS

When discussing communications, there are two basic types to be considered; man to machine and machine to machine.

1. Man to Machine

For communications with the graphics console, the light pen is considered to be one of the most effective methods. The reason being, that it not only can be used for the sketching or drawing of figures but may also be utilized to control the flow of the program. Other widely

used means of communications are by alpha-numeric keyboard, function switches, control dials, sketch tablet, joystick and mouse. The joystick, mouse and variable control dials are analog devices which require the presence of a digital to analog converter [11].

2. Machine to Machine

In the case of a shared system (i.e., a graphic console and large host computer) there must be a relatively simple high speed interface between the two processors. It should be capable of rapidly transferring large quantities of information between the two machines.

IV. DESIRABLE CHARACTERISTICS OF AN INTERACTIVE GRAPHICS SYSTEM

A. OVERALL CHARACTERISTICS

1. Cost Effectiveness and Response Time

A primary consideration in the selection of an interactive graphics system, as with the selection of any expensive computing hardware, is the cost effectiveness of the system. To this end, the decision as to whether to obtain a medium sized computer, which is strictly dedicated to graphics or obtain a large time-shared system consisting of a small computer linked to a larger one becomes increasingly important.

When arriving at this decision it must be determined whether a strictly graphical system would be utilized enough to be economically feasible. In most cases the answer to this question would be no, and a system which allows for the host computer to be utilized for "batch-processing" when not needed for graphics applications should be chosen. In addition, the system should provide rapid response when in the shared mode, as a user who thinks nothing of waiting overnight for a "batch-job" becomes impatient when having to wait much over 20 seconds for a reply when seated at a console [12].

2. Simplicity and Effectiveness

In addition to being cost effective and having a quick response time, "two properties of an interactive system which are essential to the usefulness of the system, but which compete with each other to some extent are effectiveness and simplicity of use" [13].

Effectiveness is defined as the ability to solve significant problems in a reasonable amount of time and simplicity is merely the

quantity and complexity of actions required by the user. It is easy to see how these two could be in conflict, as generally speaking the more complicated a system is made the more difficult it is to operate. An excellent method of testing a system for simplicity of learning is to observe the relative ease with which a user is able to operate the system. A similar test for simplicity of use is to observe the number of actions and the difficulty with which they are carried out.

B. GRAPHICAL CHARACTERISTICS

In order for man and machine to work in a close partnership with each other, the computer should have the capability to accept, interpret and remember not only shapes, but any descriptive information which might be introduced graphically [14].

1. Draw/Sketch Capability

To fulfill the aforementioned requirement the user should be able to introduce geometrical figures into the machine by specifying the points to be connected and then invoking a draw command to connect the points or by selecting a sketch function, allowing the user to draw the figure with one continuous line. This would be done by a light pen or other similar input device.

2. Editing Capabilities

Routines which modify the geometrical representation of the figure, such as scaling up or down, or changing the display from a dashed picture to solid, as well as routines to rotate or translate the figure in order to obtain different views are especially useful. In addition, a set of functions available to produce the more common geometrical shapes such as circle or square saves much programmer time and effort.

3. Speed

One of the main advantages in using a digital computer is the high speed obtained in making routine calculations. Speed should also be a factor in our graphics system. However, a highly interactive graphics system will not only do calculations rapidly, but will also allow the user to view immediately any modifications which are made to the display.

4. Miscellaneous

While most of the graphical applications mentioned to this point may adequately be displayed on a two dimensional system, a console which has the added feature of a three dimensional display is advantageous. This is especially true in systems utilized for any type of design applications.

Finally, in any graphics system be it of the stand alone variety or one which is linked to a large multiprocessing unit, storage space is likely to be at a premium. One, and perhaps the most efficient means of helping to minimize this problem is to have library callable subroutines. These could then be called in from secondary storage as required to perform routine operations. By using this method, only the operating system and the program concurrently running need be in main storage, saving space, while allowing the user to work with larger and more complex programs.

While these are by no means the only graphical characteristics which are considered desirable they are the most important and a majority of them should be included in any up-to-date well organized system.

C. INTERACTIVE CHARACTERISTICS

Whenever a graphics application program is written it should be remembered that individuals other than those familiar with programming may have access to the system. This being the case then, the system should be capable of allowing the program to explain itself to the user in words, as well as with pictorial representations. It should guide the user through the program, step by step, presenting the results at each decision point as well as presenting the appropriate options to be taken from that point. It should also allow decisions to be deferred by storing results from various inputs until ready to make final comparisons.

1. Communications

There are many characteristics which must be designed into a system in order to make the graphical options possible. The one single characteristic upon which all others are dependent is the ease and flexibility with which the user is able to communicate with the system.

There should be a relatively simple interface between the user and the console, allowing for rapid and effective transferral of information between the two. As stated earlier a light pen makes an excellent means of communications with a graphics system. The reason being that the light pen may serve a dual purpose. It not only can be used to draw or sketch figures as is normally done, but it may also be utilized to control the flow of the program by using the device as a "pick" and making a selection from a menu. In this manner the user has the ability to tell the program where to go and what to do as desired. Thereby, providing greater versatility in programming. Additionally, some means of textual input, such as an alpha-numeric keyboard, should

be an inherent part of the system. Another item which comes under consideration in discussing communications and is considered critical to the system, is the language selected for implementation on the machine. A purely graphical language such as POGO [15] or GSP [16] is the most desirable as it allows the user to communicate with the machine in terms that are more meaningful to graphics application. However, any higher level language, such as Fortran IV, which may be adaptable to the system is considered acceptable.

2. Error Recovery

A system which allows for the fact that errors are going to be made and provides for quick and nondestructive error recovery procedures will provide for easier and more complete interaction by the user.

3. Data Structures

The primary requirement of a graphics data structure is to link names (memory addresses) with images and sub-images. In order to contribute to the interactivity of the system the data structure should be capable of performing the following actions; the addition of new or the deletion of old information, to search, sort or duplicate the information stored in the system and finally, it should allow for the resequencing of the data structure and reallocation of memory once these actions have been performed. (This procedure is commonly referred to as garbage collection).

There are three basic types of data structures used in computers today [17]; the sequential, random and list.

a. Sequential Structure

The sequential structure is the simplest and is merely the storing of records [17] sequentially in memory according to a common

attribute. This has the advantage of allowing records to be retrieved quickly while having the disadvantage of being slow to insert a new or delete an old record because the whole file must be updated whenever a change occurs.

b. Random Structure

A random data structure utilizes a bookkeeping procedure which assigns an arbitrary address to a piece of data when it is received and all references made to the entry are by that address thenceforth. Other means of using a random data structure are by hashcoding [18] and Table lookup [18].

In hashcoding the name itself is treated as a number and standard calculations are performed on it giving an address. In the table lookup method, the most popular random method, a table of names and addresses is set up and each time a reference to a particular block is made, the address is looked up in the table.

c. List Structure

The main characteristic of a list structure is that the records are chained together by a series of pointers. A pointer being simply a word in memory which contains an address of a word or series of words in memory. There are many different kinds of list structures; linear, circular, doubly linked and orthogonal [19], to mention only the most familiar. The most commonly used of these is a circular list called a ring structure. A ring structure is one that has the last address being a pointer to the head of the list. The advantage of a ring structure is that it is very easy to work with. To enter a new element, all that is required is to put the element in memory and have a bookkeeping procedure which ensures the connection of the forward

and backwards pointers. For deletion of an element the reverse procedure is followed.

The utilization of most any of the data structures discussed would be satisfactory for an interactive graphics system. However, the most efficient is one which utilizes a combination of two types, using one for the drawing functions and another for operating on the completed drawings [20].

V. PRESENT SYSTEM

A. HARDWARE DESCRIPTION

The Computer Laboratory's graphical display system is composed of four main pieces of hardware; a general purpose digital computer (the XDS 9300), an analog computer, the Ci 5000 (not a topic of discussion in this study), and two small graphics computers produced by the ADAGE Corporation. (Hereafter referred to as either ADAGE or AGT/10). The hardware configuration along with the various communications paths are shown in Figure 1.

1. XDS 9300

The XDS 9300 [21] may be utilized by itself to process XDS Fortran jobs or it may be used as the host computer for the graphics consoles. It has a 24 bit word, 32K words of main storage and a magnetic drum that may be used for secondary storage. Control of the 9300 is by commands issued at the operator's console or by inputs to the teletypewriter, card reader, paper tape reader, line printer and two magnetic tape drivers. Due to the fact that the 9300 has an interrupt driven real time capability, a sophisticated CPU capable of handling a large instruction set, and the added capability of referencing a large secondary storage device, it is especially useful in graphical applications.

2. Adage AGT/10

The two Adage AGT/10's may be operated independently, in the stand alone mode, or in conjunction with the XDS 9300 host computer. The Adage graphics terminals are display stations consisting of the DPR

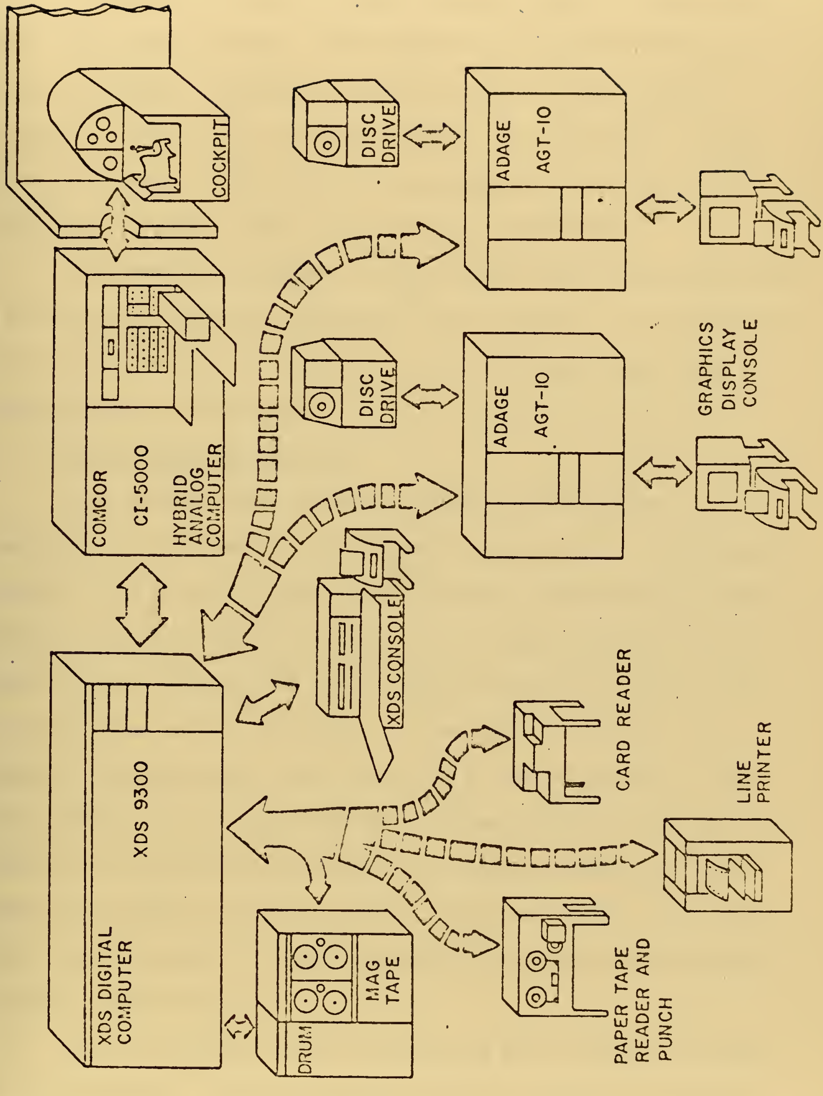

COCKPIT
COMCOR
CI-5000
HYBRID ANALOG COMPUTER
DISC DRIVE
ADAGE
AGT-10
DISC DRIVE
ADAGE
AGT-10
GRAPHICS DISPLAY CONSOLE
XDS CONSOLE
XDS DIGITAL COMPUTER
XDS 9300
CARD READER
LINE PRINTER
PAPER TAPE READER AND PUNCH
MAG TAPE
DRUM

Figure 1

2 digital processor, a vector generator and a character generator [22].

The DPR 2 is a general purpose digital computer having a 30 bit word, 8K of primary storage, and a magnetic disk pack available for secondary storage. Interactive devices supplied with the system include: a light pen, 16 function switches, two foot pedals, a joystick, 6 variable control dials and an alpha-numeric keyboard. The only other peripheral device is a paper tape reader/punch.

The character generator is a high speed stroke writer [22] with a 64 character vocabulary, expandable to 96 characters. The characters can be displayed in one of three sizes at one of three intensities and either upright or italicized.

3. XDS 9300/AGT/10 Interface

The communications between the AGT/10 and XDS 9300 is conducted through a sophisticated interface system [23]. The interface provides for a 24 bit data path which transfers information at a rate of 250 KHZ to the 9300 while allowing for a transfer rate of 160 KHZ from the 9300. Only 24 bits are transferred because this is the size of the computer word in the 9300. In transferring information it should be remembered that the first 6 bits of the AGT/10 word will be lost, therefore a shifting operation, is normally done on this word to obtain the proper degree of significance, prior to sending the information to the 9300. Conversely, when information coming into the AGT/10 has been received it is usually shifted in the opposite direction to fill up the first six bits.

The data transfer is half duplex [23] and is always initiated by an AGT program. Transfer can occur between any AGT core location and any XDS location in excess of 2000_8, the maximum length of a block

transferred is 8K, core size of the AGT/10

B. GATED

1. Description

GATED is a graphics and text editor program written in ADEPT, the AGT/10 assembly language, and is loaded from disk into main memory, by the system monitor, upon command. It may be used in the stand alone mode to draw or sketch pictures, but it is required whenever the Adage is used as a terminal in conjunction with the XDS 9300.

The purpose of GATED is threefold:

1. It is the means of communication (i.e., the interface) between the graphics console and the operator. It has the capability in the stand alone mode of accepting graphical input from the light pen, joystick, foot pedals, variable control dials and function switches, while obtaining textual inputs from the keyboard. When used in conjunction with the 9300 the light pen is the primary interface for graphics, while again the keyboard is used for text editing. By utilizing these devices properly, data may either be created or existing data may be edited.

2. The second purpose GATED serves is to communicate with the Fortran callable graphics programs and subroutines which the user may store in the XDS 9300. Through the interface discussed earlier the Fortran program may create and edit data displayed on the AGT/10 CRT.

3. Finally, in addition to the normal editing operations carried on by GATED, the display screen is refreshed constantly at 40 frames per second. It is noted that refreshing of the picture discontinues during data transfers by the interface between the AGT/10 and the XDS 9300.

2. Data Structure

GATED allocates storage for graphics and text data individually. It further divides each into blocks [21] and utilizes separate bookkeeping procedures to keep the two from being confused.

Text data is stored as alpha-numeric strings each block consisting of 26 words in memory. The first two words contain control

information and the remaining 24 words are filled with 4 characters per word. This gives a block length of 96 characters, allowing for one text line per block which is equivalent to one line on the display screen. This allows for a maximum of 40 text lines to be displayed. It is noted that when transmitting text from the 9300, it may only be sent one block (i.e., one line) at a time, and then it must be specified by line number and character position.

Graphical displays are represented as a series of straight lines. Each graphics block consists of one word of control information followed by the number of data words required to draw the picture, stored sequentially in memory. The first word of a graphics block consists of 14 empty bits followed by 1 bit which determines whether the figure being drawn will be dashed or solid. The last 15 bits are used to specify the intensity of the picture, the last bit of which is always zero.

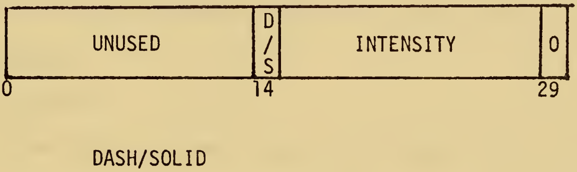


Figure 2
(AGT/10 Graphics Control Word)

The second and succeeding data words contain the x, y, cartesian co-ordinates for the data point that the word represents, as well as a bit which determines whether the point is a move or a draw, and finally a bit which indicates end of vector. The end of vector bit is always a zero except for the last word which is a one, indicating to the vector generator that it is finished drawing.

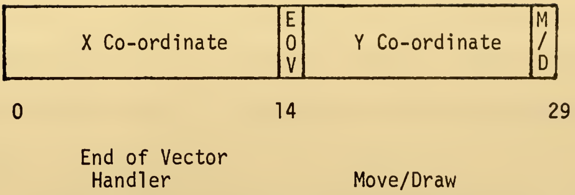


Figure 3
(AGT/10 Graphics Data Word)

There is a simple garbage collection routine used by GATED to keep the graphics block compact, thereby conserving storage space. This routine is called SHRNK and is invoked anytime a graphics block is created at the AGT/10 or sent over from the 9300. Whenever a block is selected, SHRNK searches it to ensure there are not two or more data words which contain moves stored sequentially in memory. If in fact there are two or more in a row the SHRNK routine eliminates all but the last and shrinks the block in such a manner that it still resides in sequential memory locations. The space from which the words have been deleted are then returned to memory for reallocation.

3. Operation

a. General

Once GATED has been loaded into memory from the secondary disk storage and executed by typing GATED!, C/R (where C/R stands for carriage return), or GATED(1)!, C/R, depending on whether the AGT/10 is being used with the 9300 or in the stand alone mode, the user has a number of options available.

By utilizing the light pen, function switches and teletypewriter the operator has the capability of creating or editing both text and graphics blocks as desired.

The function switches provide for the master control of

GATED. As seen in Figure 4 they allow for the selection of the desired mode of operation as well as making available a limited number of editing functions such as: erase, move, draw, and track-end point operation.

In the track-end point mode the end point of a vector, or the intersection of two vectors may be selected by the use of the light pen. This point will then become attached to the cursor which is displayed and follow the light pen as long as the track-end point function switch is depressed.

b. Graphics Editing Procedure

In order to draw a picture on the display tube, the light pen must first be turned on and the following sequence of operations carried out.

1. Depress function switch 2 in order to put GATED in the GRAPHICS SELECT mode.
2. By utilizing function switches 5, 6 and 7 (see overlay Figure 4 for specific function of each switch) the graphics block which is to be edited may be selected.
3. Depress the GO EDIT button, function switch 8. This now puts the system into the GRAPHICS EDIT mode. The creation of the new data may be thought of as editing a new block.
4. At this point the cursor may be turned on, by depressing function switch 9, and the block edited by use of the light pen.
5. While editing the display, function switches 13, 14, 15, and 16 may be utilized, giving the capabilities of move, draw, sketch and erase respectively.
6. By depressing END-EDIT, function switch 4, control is returned to the 9300 if in that mode of operation or to GATED itself if operating stand alone. In addition, the garbage collecting routine, SHRNK, is called and the unused portion of memory is returned to the machine for further allocation. This is made necessary by the fact that if a new block is selected, the entire remaining AGT/10 core is made available to the operator.

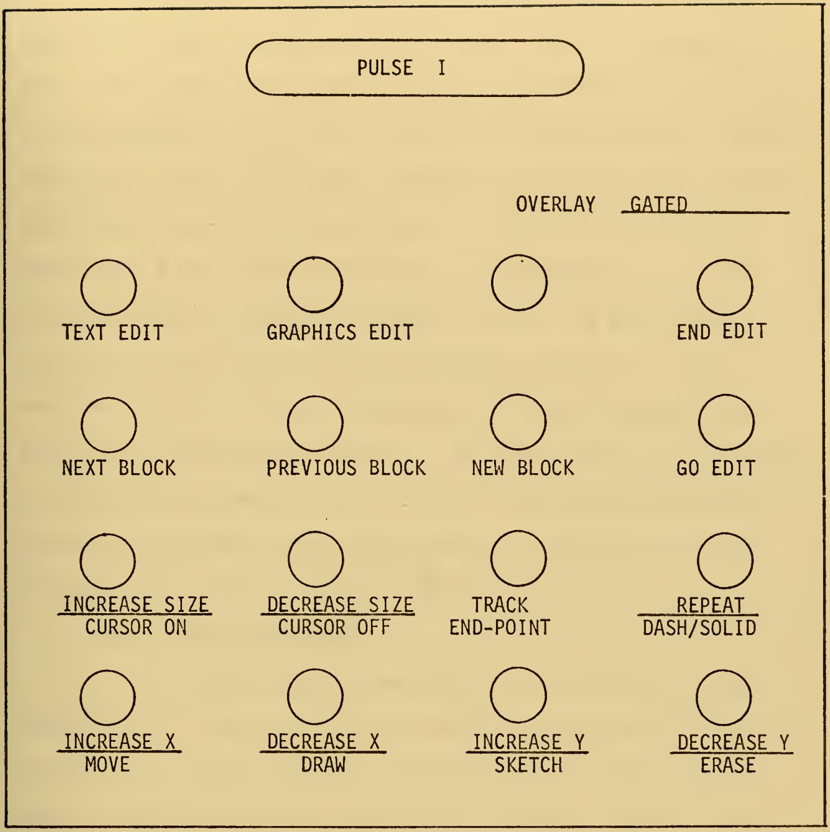


Figure 4
Function Switches
(GATED Overlay Shown)

c. Text Editing Procedure

Text edit procedures are similar to those of the graphics edit mode. Once in the text edit mode and a block is selected, a cursor will appear on the screen and may be positioned as desired by the teletypewriter. By utilizing the alpha-numeric keyboard, desired text input is made. The size or position of the text may be further modified by using function switches 9, 10, 12, 13, 14, 15, and 16 (see Figure 4 for specific operations). By depressing one of the switches, marked increase or decrease (x or y), the position of the text will be moved one line or one character position at a time. To move the text more than one position at a time the desired function switch may be depressed repeatedly or the same effect may be obtained by depressing the desired function switch and REPEAT simultaneously. Pressing the END EDIT switch causes control to return in the same manner as discussed under the graphics edit operation.

d. Fortran Operations

To support the Fortran operations performed by the XDS 9300 there are a number of library callable subroutines which are required for the smooth operation and interface of the two machines. These will be covered in the discussion of RADIK as many had to be changed to obtain the desired results.

C. SYSTEM EVALUATION

The resident computer graphics system in the Computer Laboratory is adequate for most applications and demands which are placed on it. However, there are limitations in the present configuration and improvements to be made. This section is an attempt to objectively identify and discuss these limitations.

Because GATED is the primary interface between the operator and the graphics console the greatest amount of attention was focused in this area. However, a cursory look was made into the present hardware set up and a few deficiencies were noted.

1. Hardware Limitations

At this time the Adage graphics terminals are capable of communicating with the 9300 in series, i.e., one at a time. While this is not considered a serious restriction on the system, the fact that only one terminal at a time may utilize the XDS 9300 to host a Fortran program is considered unsatisfactory.

There is much computer time lost due to the fact that an AGT remains idle merely because the present system only allows one Adage to be interfaced with the 9300. With the multitasking and multiprogramming capability possessed by the 9300 there should be a means by which both terminals access the 9300 simultaneously, through a system of priority interrupts similar to an actual time-sharing system. This would in fact make the system more cost effective through reduction of dead time.

While not completely a hardware problem another deficiency which exists is the difficulty that arises when trying to display text. This is especially true when operating stand alone, utilizing some program other than GATED. At present it is extremely difficult to input or edit a textual display, produced in conjunction with a figure. Part of the problem lies in the character generator and the refreshing of the picture, while the other is the manner in which the character generator is invoked. By upgrading the text facilities of the system, a greater versatility in programming could be achieved.

2. GATED Limitations

The program GATED is a simple and effective interface between the operator and the machine. While it adequately handles the tasks which have been programmed into it, there have been omissions which limit the systems capabilities, whether in the stand alone mode or in operations with the 9300. The majority of these limitations may be eliminated through programming expertise as the hardware capability is already present in the equipment at hand.

a. Flexibility of Communications

The first of these programming oversights for which there is hardware readily available is flexibility of communications. GATED is controlled by the function switches and accepts inputs from the light pen and teletypewriter. However, this still leaves two unused sources of communications, the joystick and variable control dials.

b. Control of XDS 9300

By proper programming the increased flexibility of communications could help to overcome the fact that an operator has very little control over the XDS 9300 when at the graphics console. At this time the only control, which the user is able to exert over the flow of a Fortran program, resident in the 9300, is pushing the END EDIT function switch which causes the program to return to the point of the edit call and continue execution. This is a severe limitation and causes much consternation on the part of the user.

c. Graphics Editing Functions

While the hardware will allow it, at present, the system has no built-in editing functions which gives the user the capability of scaling or moving the display about the scope in order to obtain a

different perspective of the figure. In addition, there are no built-in functions which allow for the drawing of common geometrical figures such as circles or squares and it is a waste of the operator's time to have to construct these each time one of the figures is desired.

d. Text Editing Functions

An option similar to the one discussed in the previous paragraph is considered desirable in the text edit mode also. Even though the routines are available which allow for the moving of text data around the screen, there is no capability for transmitting the new position back to the XDS 9300. This is because the bookkeeping procedure utilized in keeping track of the text blocks by the AGT and 9300 are different. The AGT keeps track of blocks only and the 9300 attaches a line number and character position to each block of text and it is referenced in this manner.

3. Fortran Limitations

In addition to the limitations placed on the system by GATED there are some changes which could be implemented on the 9300 "side of the house" that would enhance the versatility and efficiency of the system.

a. Data Transfer Inefficiency

The first of these would be to change the procedure by which graphics blocks are sent to the 9300 from the AGT. Now, the size of the block sent back to reside in the 9300 core is strictly dependent on the value to which the block was dimensioned in the initialization steps of the Fortran program. This occurs independent of the number of words required by the AGT. It would be much more efficient to have the AGT return only the number of words required

for the display and invoke a garbage collection routine similar to the SHRNK routine available to GATED for the return of core for reallocation.

Another problem similar to this is the fact that all graphics blocks must be transmitted from the 9300 sequentially starting with number one. It would be much more convenient if there was an option which allowed for the block to be referenced by name or in any numerical order, designated by the operator.

There are, to be sure, other items which contribute to the ineffectiveness and impose limitations on the system. However, these are considered to be of major importance and will be discussed in greater detail in the section on RADIK.

VI. RADIK

A. GENERAL DESCRIPTION AND PURPOSE

RADIK is a graphics and text editor program developed in association with Lieutenant R.F. Ashford, Jr. for implementation on the AGT/10, Adage graphics terminal. It is written in ADEPT, the AGT/10 assembly language, and is a direct descendent of GATED; the graphics and text editor in use at this time.

RADIK was developed in an attempt to overcome the previously discussed limitations of the graphics and text editor program presently used in the computer graphics system. By the very nature of the work required for the accomplishment of this task, the problem had two easily definable areas of responsibility. The first, which came under the cognizance of Lieutenant Ashford, was the development and implementation of routines for the AGT/10 graphics terminal. In order for these to function properly and with any degree of sophistication the supporting subroutines for the XDS 9300 graphics package had to be developed. This aspect of the project came under the responsibility of the author.

Throughout the development of RADIK a high degree of interaction and a free interchange of ideas were required to ensure the successful completion of the project. In many areas, as in the change required to the data structure, the work overlapped and the highest degree of coordination of effort was required in order to obtain the desired results.

RADIK, the final outcome of this joint effort, has the following purposes:

1. Act as a highly interactive interface between the user and AGT/10 graphics terminal, allowing for full utilization of all peripheral hardware.

2. To serve as a method of controlling and communicating with the Fortran callable graphics programs and subroutines resident in the XDS 9300.

3. To not only refresh the picture as done by GATED, but to also provide additional editing capabilities to supplement the routines presently available. By utilizing the function switches in the edit mode, the following operations may be performed on the display:

    a. Translation in the x and y planes,

    b. Scale up or down, and

    c. Rotation of the display.

B. DEFINITION OF THE PROBLEM

The ultimate goal in the development of RADIK was to produce a highly interactive graphics and text editor program which not only possessed all of the capabilities of its predecessor, GATED, but would accomplish the following objectives as well:

1. Provide for the inclusion of scale factor as an argument to the Fortran callable subroutines utilized for graphical applications.

2. Provide for the inclusion of DX and DY as arguments to the Fortran callable subroutines utilized for graphical applications.

3. Development of an additional graphics subroutine which would allow combinations of scale factor, intensity, DX, DY, and dashed/solid information without sending the x/y data list.

4. To provide editing routines which allow for the rotation, translation and scaling of a display by utilizing the function switches.

5. Allow for specification of graphics blocks by name or number, without being restricted to transmitting them in sequence.

6. To provide for the sampling of function switches, joystick and variable control dials by Fortran callable subroutines.

7. Elimination of the problem whereby GATED is "locked" into the edit mode, (i.e., no control over the program from the graphics console).

8. Make the graphics Fortran package more efficient by modifying it in such a manner, so as to store the "current length" of graphics blocks returned from the AGT/10.

9. Allow for the return of text to the 9300 which has had its position modified at the AGT/10.

C. PROCEDURE FOLLOWED

In the development of each of these objectives there were two main requirements to be met. First, the affected areas in GATED had to be found and the ADEPT coding modified. Secondly, the Fortran callable subroutines required for the support of graphics programs in the 9300 had to be modified or in some instances new ones introduced. These were written either in meta-symbol, the 9300 assembly language or Fortran.

1. Translation and Scale Factor

In attacking the problem the decision was made to first construct a new data structure and use this for the basis on which to build the other changes. Because the first three objectives were closely related to the data structure problem, they were attempted simultaneously.

a. Additions to RADIK

(1) Data Structure. The data structure of RADIK is similar to that of GATED in that storage is still allocated separately for graphics and text data.

Furthermore, graphical displays continue to be represented as a series of straight lines. However, the internal representation has changed somewhat. The first control word of a

graphics block consists of 14 bits allocated to scale factor, followed by 1 bit which advises the machine whether to display the image as a solid or dashed line. The first 15 bits contain the DX value and the last 15 bits the DY value. The 15th and 30th bits are always zero because they are the end of vector and move/draw bits respectively. If they were not zeroed out, the vector generator would consider itself finished drawing the figure after the second word and if the draw bit was on, a line would be drawn from the center of the scope to the point DX, DY which would be unsatisfactory.

The combined structure of the first two control words is as shown in figure 5.

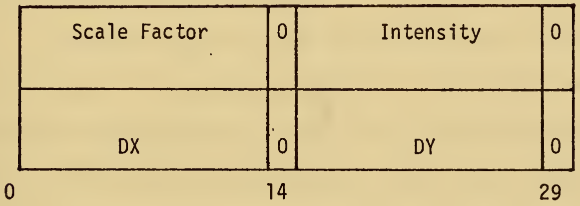


Figure 5
(First two control words of a graphics block)

The remaining data words are set up in the same type structure as is found in GATED.

2. <u>Other Changes</u>. Due to the fact that GATED already had the proper equations available

$$X' = DX + Sc(X)$$

$$Y' = DY + Sc(Y)$$

Where: X' is the new value of X.

Y' is the new value of Y.

DX is the value of the X displacement.

DY is the value of the Y displacement.

Sc is the value of the scale factor.

The other major additions to RADIK were in reshuffling the many pointers to the graphics blocks to ensure the control words were accessed in the proper manner and the changing of the SHRNK routine.

SHRNK, the garbage collecting routine previously accessed by GATED required modification in order to accept the first two, and possibly even the third word as a move. Prior to changing SHRNK, the routine was eliminating the first word, using the second word and using the first data point (3rd word) as DX and DY. This resulted in the distortion of the display.

b. Fortran Routine Required

In order to operate with RADIK's new data structure there had to be a change in the method in which the words were constructed in the 9300 for transmittal to the AGT/10. Formerly this was accomplished by invoking the on-line function IHEAD [22], which through a series of shifting and masking operations would place the calling arguments; dash/solid and intensity, into the proper format as shown in figure 2. A new function, IPAKM (see appendix for listing), which includes scale factor as a calling argument was developed. In the routine, IPAKM is given the value of IHEAD (dash/solid, intensity) logical OR'ed with scale factor. This results in the control word of figure 5 (page 39).

In addition, a new subroutine UNPAKM, to unpack the first control word when transmitting edited data from the 9300, was required. The routine is called with the first control word and the variable names for intensity, dash/solid and scale factor. In the routine a

series of ANDing operations occur which stores the proper values into intensity, dash/solid and scale factor for storage in the 9300.

c. Applications and Limitations

By utilizing these routines it is no longer necessary to transmit the X/Y data list in order to translate or scale the figure, as was previously required. It is only necessary to pass the first two control words, once the X/Y data list has been stored in the AGT/10. Additionally, this gives the capability of being able to pass the parameters for translation and scale factor at the time of a subroutine call to output a graphical display from the 9300.

When utilizing these routines, DX and DY may take on any value between +1.0 and -1.0, while scale factor may range from 0, a point, to 3776_8. The limitation of 3776_8 is placed on scale factor because there are only 12 bits allocated for this parameter and if any number larger than a 3 was utilized in the first digit it would result in the first bit being a 1 and the scale factor would be negative, which is undefined. Additionally, with the restriction that the last bit be a 0, because of the end of list vector, a number which ensures the last bit is a 0 (0, 2, 4, 5, or 6) must be utilized in the last digit. It is noted that some capability of the AGT/10 is lost as it will handle a scale factor of 37777_8 which is limited by the length of the line the character generator is capable of producing.

d. Operation

In order to invoke the routine described, the normal graphics editing routines are utilized, substituting IPAKM for IHEAD and UNPAKM for UNPACK on the first data word only. A typical call to IPAKM would be as shown on the following page:

IMAGE (1) = IPAKM (0, 10, scale)

Where;

IMAGE (1) is the first control word of the display.

0 is the dash/solid parameter, in this case a solid figure will be drawn.

10 is the intensity at which the figure is to be displayed.

Scale is the variable to which the desired scale factor is assigned;

A typical call to UNPAKM would be:

CALL UNPAKM(FRAME(1), INT, IDSH, SCALE)

Where;

FRAME (1) is the same as IMAGE(1).

INT is the variable name in which intensity is stored.

IDSH is the variable name in which dash/solid information is stored.

SCALE is the variable name in which the scale factor is stored.

2. Editing Routines

The next problem attacked was the development of routines, in RADIK, to allow for the translation, rotation and scaling of a figure while seated at the graphics console with the system in the GRAPHICS EDIT mode.

a. Additions to RADIK

To obtain the desired result it was necessary to add seven additional sets of code to allow for: translation; up, down, left and right, scale; up or down and counter clockwise rotation. The code which was developed is shown in the Computer Program. In addition it was necessary to insert coding which would allow for function switches 1, 2, 3, 5, 6, 7, and 8 to be utilized, by the applications program after the GO EDIT function switch has been depressed.

b. Fortran Routines Required

It was not necessary to introduce any new Fortran procedures in order to make these operations possible as the Fortran routines normally utilized for graphical operations are quite adequate in this case.

c. Applications and Limitations

It is considered advantageous for a user to be able to translate, rotate and change the size of a picture while seated at the graphics console. By having this capability the user may transmit a rough figure from the 9300 and by entering the GRAPHICS EDIT mode fine tune the display. Additionally, this provides the ability to obtain different views of a display.

There are two limitations built into these routines neither of which is considered a serious detriment to the system. The first of these is that if a figure is input from the 9300, or from the graphics console the first data point must not be a move. If a move is generated and eventually rotation is desired the display will become distorted while rotating. This may be avoided and the desired results obtained by merely slowing the cursor to the desired point of the move and start drawing from there. The other limitation is that the figure may not be rotated for an excessive number of revolutions as the picture will become smaller. The reason for this is because of the manner in which the arithmetic is done on the array containing the data points. In order to prevent this it would be necessary to have a dummy array of some standard length saved aside to store the data list in, while the numerical operations are performed. It is not considered that the effort required to accomplish this justifies the

results which would be obtained, especially since the scale factor routine may be utilized to obtain the proper size picture.

d. Operation

In order to operate these functions it is necessary to depress the GO EDIT function switch (number 8) once the desired graphics block has been obtained. At this time function switches 1, 2, 3, 5, 6, 7, and 8 are freed from their normal duties and given the capabilities as shown in the RADIK overlay, Figure 6. The function switches remain available until the END EDIT function switch (number 4) is depressed.

3. Sampling of Function Switches, Joystick, and Variable Control Dials

By devising a method by which the function switches, joystick and variable control dials may be sampled, on command from a Fortran program the flexibility of communications with the system is again improved.

a. Additions to RADIK

In order to allow for the utilization of the previously mentioned hardware it was necessary to program RADIK to first recognize a new special command, for each individual piece of hardware which could be accessed from the 9300. This in turn causes RADIK to make an external call on TRJSB [24] and TRVCD [25] in order to obtain the present values of the joystick and variable control dials respectively. In addition it was necessary to include code which would send an interrupt to the 9300 to allow for the data to be transmitted, and then signal completion of the action. It was relatively easy to obtain the values of the function switches as this information was already being stored by RADIK in the normal course of events.

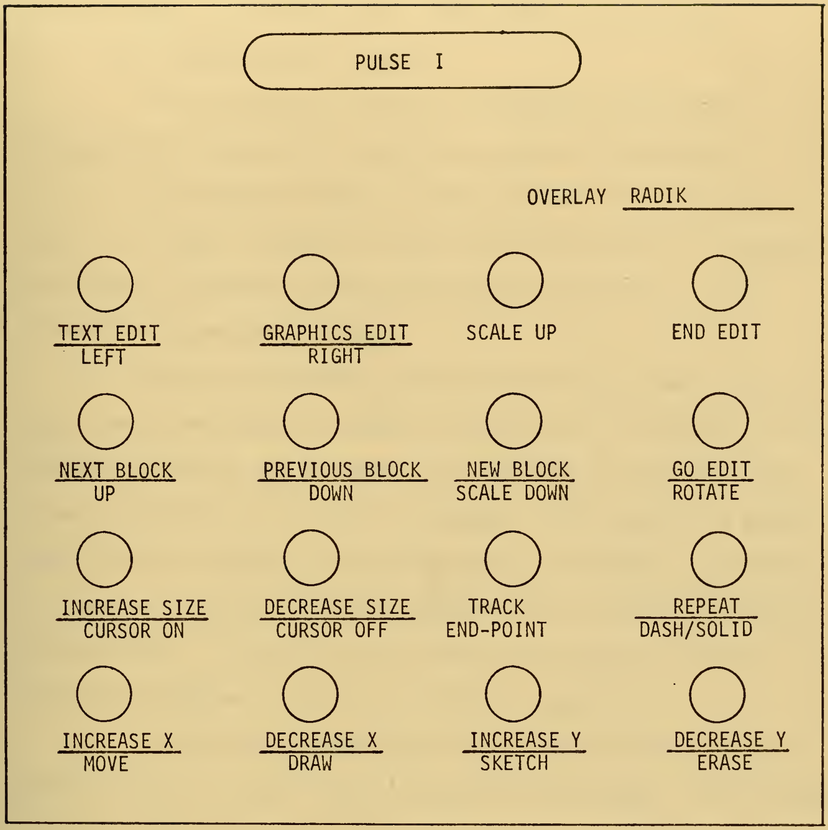


Figure 6
Function Switches
(RADIK Overlay Shown)

Once the values were obtained they could be transmitted to the 9300 in the same manner as for the joystick and variable control dials.

b. Fortran Routines Required

For each piece of hardware a metasymbol routine was developed for inclusion into the library package of subroutines.

The purpose of each of these routines, VCD, JOYSTK, and SWITCH is to signal the AGT/10 to sample the desired piece of hardware and transmit the address to which the desired information is to be returned for storage in the 9300.

The values returned by VCD and JOYSTK are signed fractional numbers which are extremely awkward to operate with. Therefore, it was necessary to develop the metasymbol routine XFORM, which converts the signed fractional values into a real number. This results in a normal decimal value on which it is easy to perform all arithmetic operations. This routine is hidden from the programmer as it is called from within VCD and JOYSTK. A flow chart and listing may be found for each of these three routines as well as for SWITCH in Appendix B.

In order to determine which of the function switches were being depressed, a Fortran logical function was developed. The function is called CHANGE and requires the switch number to be tested and address in which the switches are stored to be passed as calling arguments to the function. Through a shifting and masking operation the function determines whether or not the switch to be tested is on or not. If it is on a value of TRUE is returned to the call, otherwise it remains FALSE. The routine is listed in Appendix A.

c. Applications and Limitations

By proper utilization of the routines VCD, JOYSTK, and

SWITCH the user has an excellent method of controlling the flow of a Fortran program, from the graphics console. This adds a much needed capability and upgrades the system by allowing for greater user interaction. One use to which this could be applied would be in the debugging phase of a programs development. The operator could have the program execute up to a desired point and then display the required figure. The program could be held in a loop in which one of these routines is invoked and when the desired value is set by the operator at the graphics console the program could be returned to start over, continue execution, terminate or jump to any desired instruction as specified by the programmer. As can be seen this also eliminates the problem whereby the user is "locked" into the EDIT mode at the AGT/10.

There are of course many other applications for which these routines may be utilized, limited only by the imagination of the user.

There are no known limitations or restrictions on the usage of these routines.

d. Operation

A complete description of VCD, JOYSTK, SWITCH and XFORM including examples of how and for what they may utilized may be found in Appendix B.

The logical routine CHANGE, which allows for the testing of the function switches, may be utilized anytime after the call to SWITCH has been made, preferably immediately thereafter. The call to the routine is made by using the form IF(CHANGE(3, ISWITCH)) XXX

Where;

The 3 is the number of the switch to be tested.

ISWITCH is the address that the switches are stored in. (This must always be an integer variable)

XXX is any executable Fortran statement.

The executable Fortran statement will be executed in the event the routine returns a value of true. Otherwise, the next executable statement below the IF CHANGE statement will be executed.

D. DISCUSSION OF RESULTS AND EXTENSIONS TO THE PROGRAM

As initially stated there were nine objectives, at which an attempt was made, in the development of RADIK. Of these, six were completed and have already been discussed, one was considered unfeasible and two are in the planning stage.

For the objectives which have been met, all Fortran routines and ADEPT coding is working properly and there are no known limitations other than those already discussed. The one objective abandoned as unfeasible was the one concerning the ability to send graphics blocks to the AGT/10 other than sequentially starting with the number 1. It was found in order to keep from confusing the user that it would be necessary to modify the manner in which the block numbers are displayed on the graphics terminal and in order to do this it would require more time than the results would justify. In addition, this is not considered to be that serious of a restriction.

The remaining two not completed are the ability to transmit to the 9300 only the graphics block length required and the ability to send text blocks back to the 9300 for storage after the position has been modified at the graphics terminal. It is recommended that they be included as items to be completed in future work. In addition to these two it would be desirable to have the capability of being able to produce some of the more frequently used geometrical shapes such as a square or a circle. This could be accomplished by utilizing one of the

footpedals as a toggle to free the function switches from their present tasks. They could then be utilized for this application.

One note of caution, any changes which are made should be done so only after a close inspection of the amount of core which would be required for the operation has been made. The reason for this is that the original GATED allowed for 3100 words (as shown in figure 7) of working storage and RADIK has reduced that by 400 words. Therefore, a problem could exist in future modifications in that there would not be enough working storage available to utilize the system properly.

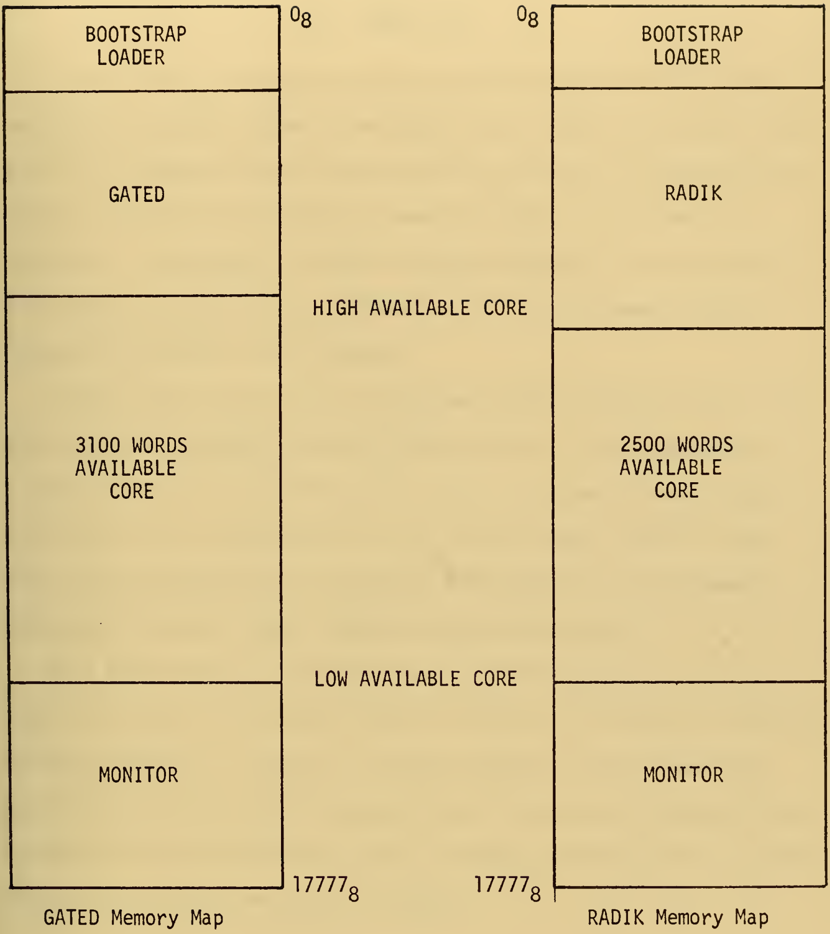


Figure 7
GATED VS RADIK Memory Map

VII. CONCLUSIONS

In the initial discussion of the desirable characteristics of an interactive graphics system it was found that interaction, that is, the ability to communicate with the system was the basic building block on which an effective and efficient computer graphics system was constructed. To this end, the theme throughout RADIK is to provide for a high degree of interaction by improving the lines of communications between the operator and the machine.

It was stated earlier that a language devoted strictly to the problem of computer graphics was the ideal approach to the programming of a system. While this in fact may be true, it has been shown in this study that a close approximation to the effectiveness gained in this manner may be approached by the proper interfacing of the equipment available, through a highly interactive operation system.

As a direct result of the programming effort expended in the production of RADIK, the computer graphics system in the Computer Laboratory has been upgraded to allow for a greater degree of user interaction. Due to this increased effectiveness the user has available the ability to better utilize the Fortran graphics package and the graphics terminal including associated peripheral hardware.

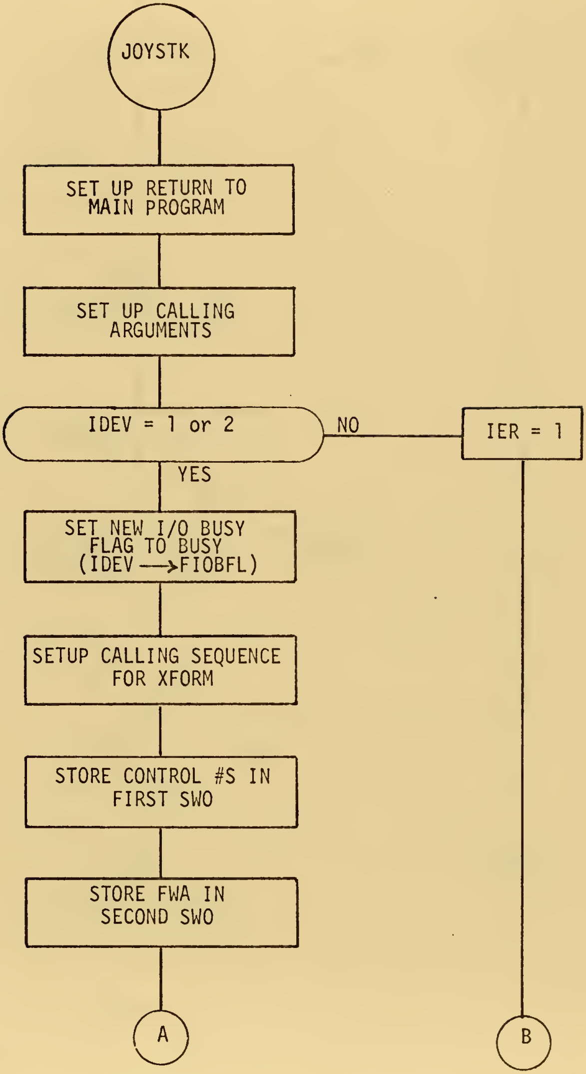
APPENDIX A

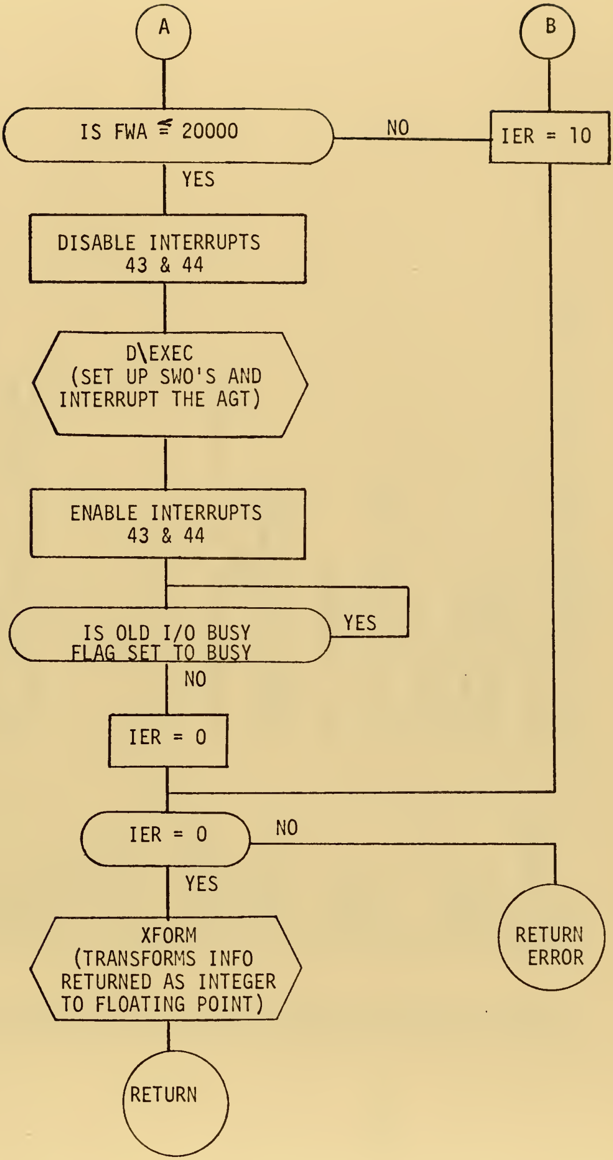
```
      FUNCTIØN IPAKM(IDSH,INT,SCALE)
      IPAKM=LIØR(IHEAD(IDSH,INT), SCALE)
  100 RETURN
      END
```

```
      SUBRØUTINE UNPAKM(IMAGE(1),INT,IDSH,SCALE)
      INTEGER TEMP
      INTEGER SCALE
      IDSH=0
      INT=10
      TEMP=LAND(IMAGE(1),00010000B)
      SCALE=LAND(IMAGE(1),77760000B)
      IF(TEMP.NE.00000000B) IDSH=1
      RETURN
      END
```

```
      LØGICAL FUNCTIØN CHANGE(Z,ISWITCH)
      INTEGER Z
      CHANGE=.FALSE.
      NØ=16-Z
      ITEMP=LCRS(ISWITCH,NØ)
      IF(LAND(ITEMP,00000001B).EQ.0) GØ TØ 50
      CHANGE=.TRUE.
   50 RETURN
      END
```

APPENDIX B


JOYSTK
SET UP RETURN TO MAIN PROGRAM
SET UP CALLING ARGUMENTS
IDEV = 1 or 2
NO
IER = 1
YES
SET NEW I/O BUSY FLAG TO BUSY (IDEV ⟶ FIOBFL)
SETUP CALLING SEQUENCE FOR XFORM
STORE CONTROL #S IN FIRST SWO
STORE FWA IN SECOND SWO
A
B


A
B
IS FWA ≥ 20000
NO
IER = 10
YES
DISABLE INTERRUPTS 43 & 44
D\EXEC (SET UP SWO'S AND INTERRUPT THE AGT)
ENABLE INTERRUPTS 43 & 44
IS OLD I/O BUSY FLAG SET TO BUSY
YES
NO
IER = 0
IER = 0
NO
YES
XFORM (TRANSFORMS INFO RETURNED AS INTEGER TO FLOATING POINT)
RETURN ERROR
RETURN

```
$JØYSTK PZE     0
        BRM     9SETUPN
        PZE     4
        PZE     0                DEV NØ
        PZE     0                FIRST WØRD ADDRESS
        PZE     0
        PZE     0                ERR FLAG
        LDA     *JØYSTK+3
        BRM     DVNØCK           IS DEV CØRRECT
        BRU     JØYDER           NØ
        STA     FIØBFL           YES, SET NEW I/Ø FLAG BUSY
        STA     JØYCL+2          STØRE IDEV FØR USE BY D/EXEC
        LDA     JØYSTK+4
        STA     JØYXFM+2
        LDA     JØYSTK+5
        STA     JØYXFM+3
        LDA     =2
        STA     JØYXFM+4
        LDA     =02000000
        STA     JØYSWØ           STØRE CØNTRØL NØ. IN SWØ1
        LDA     JØYSTK+4
        STA     JØYSWØ+1         STØRE FWA IN SWØ2
        SUB     =017777          CHECK FWA <= 20,000
        SKG     =0
        BRU     JØYMER           GØ SET ERRØR
        EØM     032004
JØYCL   BRM     D\EXEC           GRAPHIC ØUTPUT CALL
        PZE     2
        PZE     0                DEVICE NØ.
        PZE     JØYSWØ           SWØ FWA
        EØM     032001           ENABLE AGT1
        EØM     032002           ENABLE AGT2
        LDB     =0
        LDA     *JØYSTK+3        DEV NØ.
        SKA     IØBFL            LØØP UNTIL I/Ø BUSY FLAG NØT BUSY
```

```
        BRU     $-2
        CØPY    (4,5)           SET ERRØR FLAG
        BRU     JØYER
JØYDER  LDA     =1              BAD DEVICE NØ.
JØYER   STA     *JØYSTK+6
        SKU     =0
        BRU     JØYXFM
        BRR     JØYSTK          RETURN
JØYMER  LDA     =10             MEMØRY NØT ACCESIBLE BY INTERFACE
        BRU     JØYER           ERRØR EXIT
JØYXFM  BRM     XFØRM
        PZE     3
        PZE     0
        PZE     0
        PZE     0
        BRR     JØYSTK
JØYSWØ  PZE     0               ALLØCATE SWØ'S
        PZE     0
        PZE     0
        END
```

LIBRARY PROGRAM REPORT

ROUTINE NAME: JOYSTK

ROUTINE TYPE: Fortran callable subroutine

PURPOSE: To obtain the x, y and z values of the joystick attached to the AGT-10.

SOURCE LANGUAGE: Metasymbol

AUTHOR: Ralph H. Stowell, Jr.

CALLING SEQUENCE:

```
CALL (IDEV, JOY, AJOY, IER)
```

Where; IDEV = AGT being utilized.

JOY = An integer array of dimension (3) in which the digitized values of the joystick are stored.

AJOY = A real array of dimension (3) in which the equivalent floating point values are stored after being transformed in the JOYSTK routine.

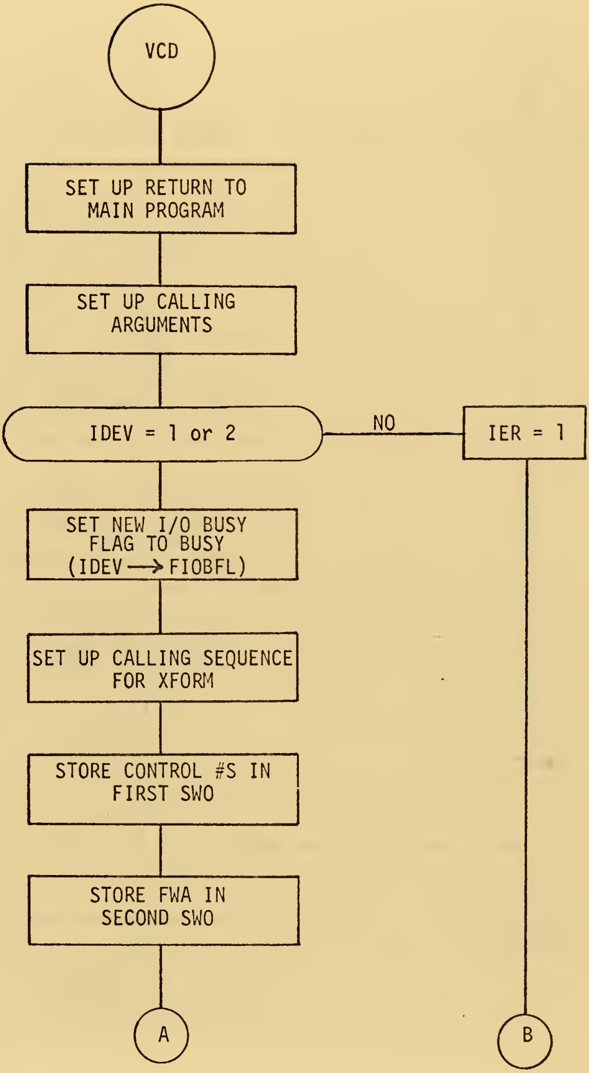
IER = Error flag.

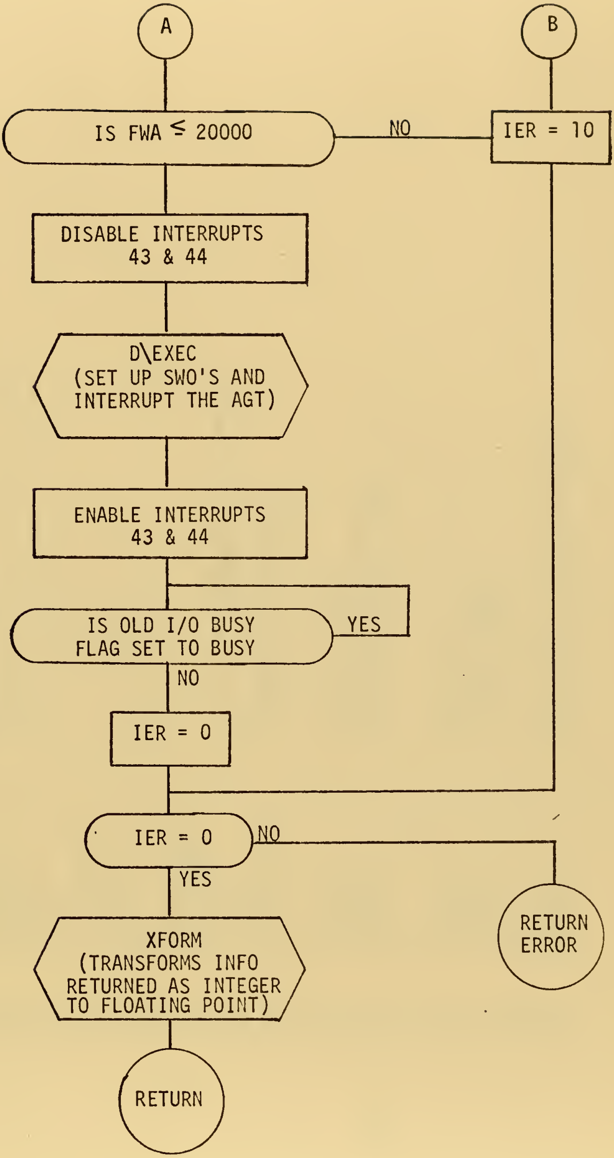
DISCUSSION: This subroutine is used to obtain the digitized values of the joystick at the AGT and through the use of another library subroutine, XFORM, transform the values into an equivalent floating point number. The user may then utilize these values to either control the flow of a fortran program or to modify a graphics block sent over from the 9300. Once the call to JOYSTK has been made the above may be accomplished similar to the examples shown below.

```
EX.     If (JOY(1) .GT. .25) stop

        DX = JOY(1)
```

NOTE: The x, y, and z values of the joystick, map to JOY(1), (2) and (3) respectively.


VCD
SET UP RETURN TO MAIN PROGRAM
SET UP CALLING ARGUMENTS
IDEV = 1 or 2
NO
IER = 1
SET NEW I/O BUSY FLAG TO BUSY (IDEV ⟶ FIOBFL)
SET UP CALLING SEQUENCE FOR XFORM
STORE CONTROL #S IN FIRST SWO
STORE FWA IN SECOND SWO
A
B


A
B
IS FWA ≤ 20000
NO
IER = 10
DISABLE INTERRUPTS 43 & 44
D\EXEC (SET UP SWO'S AND INTERRUPT THE AGT)
ENABLE INTERRUPTS 43 & 44
IS OLD I/O BUSY FLAG SET TO BUSY
YES
NO
IER = 0
IER = 0
NO
YES
XFORM (TRANSFORMS INFO RETURNED AS INTEGER TO FLOATING POINT)
RETURN
RETURN ERROR

```
$VCD     PZE     0
         BRM     9SETUPN
         PZE     4
         PZE     0
         PZE     0
         PZE     0
         PZE     0
         LDA     *VCD+3
         BRM     DVNOCK          IS DEV CORRECT
         BRU     VCDDER          NO
         STA     VCDCL+2         STORE IDEV FOR USE BY D/EXEC
         STA     FIOBFL          YES, SET NEW I/O FLAG BUSY
         LDA     VCD+4
         STA     VCDXFM+2
         LDA     VCD+5
         STA     VCDXFM+3
         LDA     =5
         STA     VCDXFM+4
         LDA     =01000000
         STA     VCDSWO          STORE CONTROL NO. IN SWO1
         LDA     VCD+4
         STA     VCDSWO+1        STORE FWA IN SWO2
         SUB     =017777         CHECK FWA <= 20,000
         SKG     =0
         BRU     VCDMER          GO SET ERROR
         EOM     032004          DISABLE BOTH AGT'S
VCDCL    BRM     D\EXEC          GRAPHIC OUTPUT CALL
         PZE     2
         PZE     0
         PZE     VCDSWO          SWO FWA
         EOM     032001          ENABLE AGT1
         EOM     032002          ENABLE AGT2
         LDB     =0
         LDA     *VCD+3          DEV NO.
         SKA     IOBFL           LOOP UNTIL I/O NOT BUSY
```

```
         BRU     $-2
         COPY    (4,5)              SET ERROR FLAG
         BRU     VCDER
VCDDER   LDA     =1                 BAD DEV NO.
VCDER    STA     *VCD+6
         SKU     =0
         BRU     VCDXFM
         BRR     VCD                RETURN
VCDMER   LDA     =10                MEMORY NOT ACCESSIBLE BY INTERFACE
         BRU     VCDER              ERROR EXIT
VCDXFM   BRM     XFORM
         PZE     3
         PZE     0
         PZE     0
         PZE     0
         BRR     VCD
VCDSWO   PZE     0                  ALLOCATE SWO
         PZE     0
         PZE     0
         END
```

LIBRARY PROGRAM REPORT

ROUTINE NAME: VCD

ROUTINE TYPE: Fortran callable subroutine

PURPOSE: To obtain the settings of the variable control dials attached to the AGT-10.

SOURCE LANGUAGE: Metasymbol

AUTHOR: Ralph H. Stowell, Jr.

CALLING SEQUENCE:

```
CALL VCD (IDEV, IDIAL, DIAL, IER)
```

Where; IDEV = AGT being utilized

IDIAL = An integer array of dimension (6) in which the digitized values of the variable control dials are stored.

DIAL = A real array of dimension (6) in which the equivalent floating point values are stored after being transformed in the VCD routine.

IER = Error flag.

DISCUSSION: This subroutine is used to obtain the digitized values of the variable control dials at the AGT and through the use of another library subroutine, XFORM, transform the values into an equivalent floating point number. The user may then use these values to either control the program flow of a Fortran program or then modify a graphics block sent over from the 9300. Once the call to VCD has been made, the above may be accomplished similar to the examples shown below.

```
EX.     IF(DIAL (1).GT.1.0) Stop

                  or

        DX = DIAL(2)
```

NOTE: The control dials are numbered vertically not horizontally as would be expected.

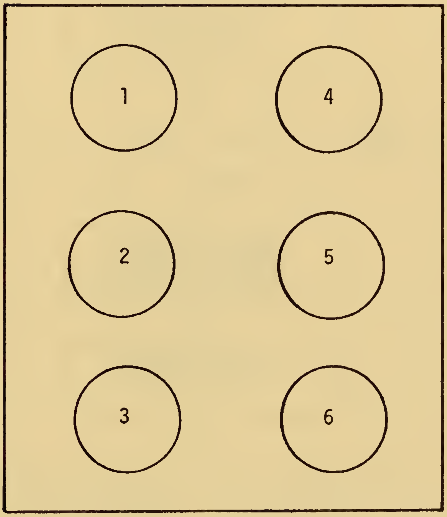
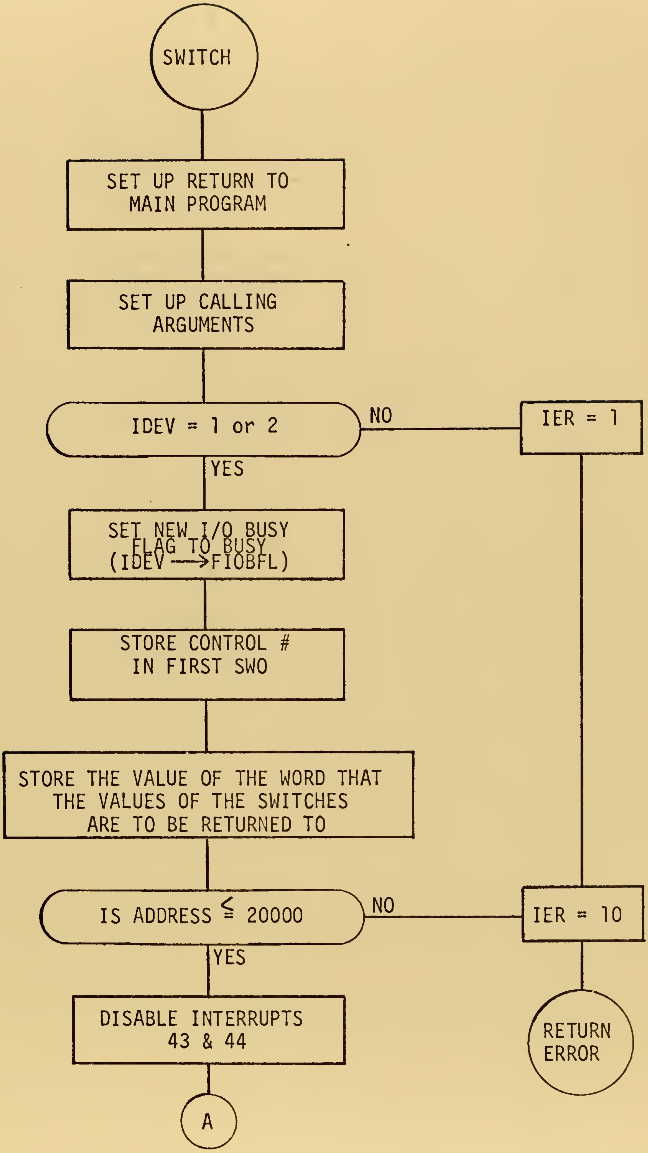
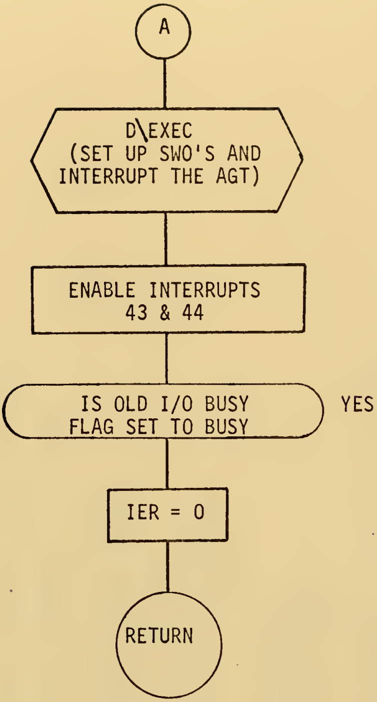


Figure 8


SWITCH
SET UP RETURN TO MAIN PROGRAM
SET UP CALLING ARGUMENTS
IDEV = 1 or 2
NO
IER = 1
YES
SET NEW I/O BUSY FLAG TO BUSY (IDEV ⟶ FIOBFL)
STORE CONTROL # IN FIRST SWO
STORE THE VALUE OF THE WORD THAT THE VALUES OF THE SWITCHES ARE TO BE RETURNED TO
IS ADDRESS ≤ 20000
NO
IER = 10
YES
DISABLE INTERRUPTS 43 & 44
RETURN ERROR
A


A
D\EXEC
(SET UP SWO'S AND
INTERRUPT THE AGT)
ENABLE INTERRUPTS
43 & 44
IS OLD I/O BUSY
FLAG SET TO BUSY
YES
IER = 0
RETURN

```
$SWITCH PZE     0
        BRM     9SETUPN
        PZE     3
        PZE     0                 DEV NØ.
        PZE     0                 ADDRESS
        PZE     0                 ERRØR FLAG
        LDA     *SWITCH+3
        BRM     DVNØCK            IS DEV CØRRECT
        BRU     SWDER             NØ
        STA     FIØBFL            YES, SET NEW I/Ø FLAG BUSY
        STA     SWCL+2            STØRE IDEV.FØR USE BY D/EXEC
        LDA     =03000000
        STA     SWSWØ             STØRE CØNTRØL NØ IN SWØ1
        LDA     SWITCH+4
        STA     SWSWØ+1           STØRE FWA IN SWØ2
        SUB     =017777           CHECK FWA <= 20,000
        SKG     =0
        BRU     SWMER             GØ SET ERRØR
        EØM     032004            DISABLE BØTH AGT
SWCL    BRM     D\EXEC            GRAPHIC ØUTPUT CALL
        PZE     2
        PZE     0                 DEVICE NØ.
        PZE SWSWØ                 SWØ FWA
        EØM     032001            ENABLE AGT1
        EØM     032002            ENABLE AGT2
        LDB     =0
        LDA     *SWITCH+3         DEV NØ.
        SKA     IØBFL             LØØP UNTIL IØB BUSY NØT BUSY
        BRU     $-2
        CØPY    (4,5)             SET ERRØR FLAG
        BRU     SWER
SWDER   LDA     =1                BAD DEVICE NØ.
SWER    STA     *SWITCH+5
        BRR     SWITCH            RETURN
SWMER   LDA     =10               MEMØRY NØT ACCESSIBLE BY INTERFACE
```

```
         BRU     SWER               ERROR EXIT
SWSWO    PZE     0                  ALLOCATE SWO'S
         PZE     0
         PZE     0
         END
```

LIBRARY PROGRAM REPORT

ROUTINE NAME: SWITCH

ROUTINE TYPE: Fortran callable subroutine

PURPOSE: To obtain the on/off value for each of the 16 function switches attached to the AGT/10.

SOURCE LANGUAGE: Metasymbol

AUTHOR: Ralph H. Stowell, Jr.

CALLING SEQUENCE:

```
CALL SWITCH (IDEV, ISWITCH, IER)
```

Where; IDEV = AGT being utilized.

ISWITCH = Location in which the returned values for the switches are returned.

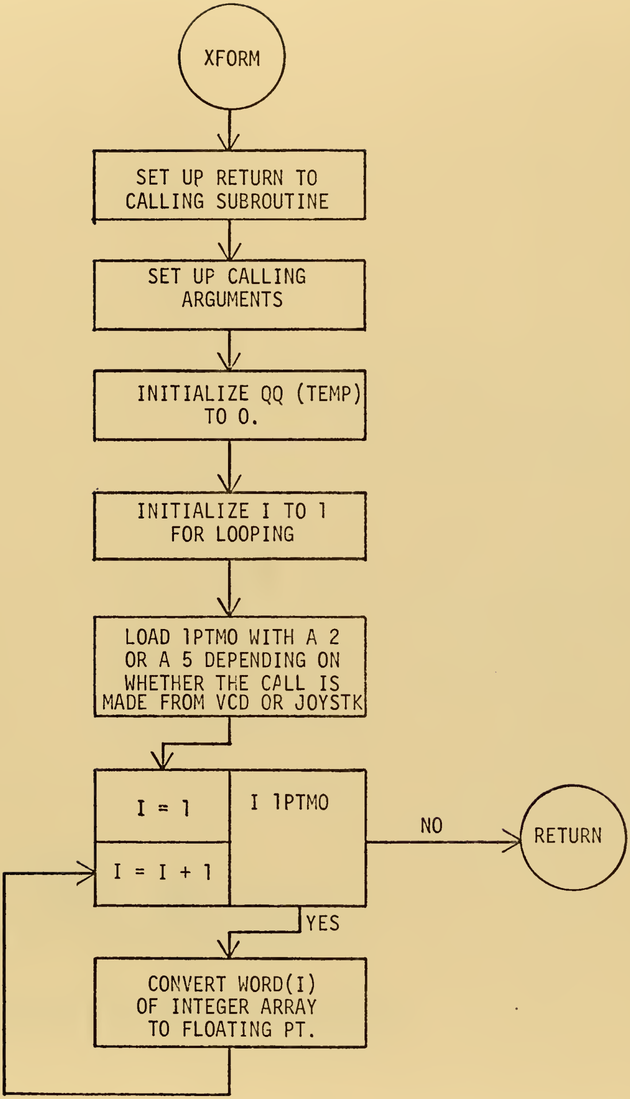
IER = Error flag.

DISCUSSION: This subroutine is used to obtain the on/off values of the function switches, which may then be used to control the program flow, or any other task the user desires. Once the call to switch has been made another routine change must be called to test the values.

EX.

```
Call switch (IDEV, ISWITCH, IER)

IF (CHANGE(3, ISWITCH)) GO to-
```


XFORM
SET UP RETURN TO CALLING SUBROUTINE
SET UP CALLING ARGUMENTS
INITIALIZE QQ (TEMP) TO 0.
INITIALIZE I TO 1 FOR LOOPING
LOAD 1PTMO WITH A 2 OR A 5 DEPENDING ON WHETHER THE CALL IS MADE FROM VCD OR JOYSTK
I = 1
I = I + 1
I 1PTMO
NO
RETURN
YES
CONVERT WORD(I) OF INTEGER ARRAY TO FLOATING PT.

```
$XFØRM   PZE     0
         BRM     9SETUPN
         PZE     3
IDUMMY   PZE     0
DUMMY    PZE     0
         PZE     0
         LDA     =0
         STA     QQ
         LDA     =1
         STA     I
         LDA      XFØRM+5
         STA      1PTMØ
XX       LDA     *IDUMMY
         LDB     =0
         FLA     QQ
         STD     *DUMMY
         MPØ     IDUMMY
         MPT     DUMMY
         MPØ     I
         SKR     1PTMØ
         BRU     XX
         BRR     XFØRM
1PTMØ    RES     1
QQ       RES      1
I        RES      1
         END
```

LIBRARY PROGRAM REPORT

ROUTINE NAME: XFORM

ROUTINE TYPE: Metasymbol callable routine

PURPOSE: To transform signed fractional values obtained from the VCD and JOYSTK routines to their equivalent floating point values.

SOURCE LANGUAGE: Metasymbol

AUTHOR: Ralph H. Stowell, Jr.

DISCUSSION: While XFORM is normally called from within VCD or JOYSTK in order to obtain the floating point value of the digitized readings returned from the AGT-10, it could just as well be called in a Fortran program to shift an integer value to floating point value. To do this the following calling sequence would have to be used.

```
CALL XFORM (INT, REAL, NO)
```

Where; INT = Integer involved.

REAL = REAL variable, the value is to be stored in.

NO = Total number of words to be transformed -1.

```
[ USNPGS GRAPHICS/TEXT EDITOR
[ H.D.K., E.L.B.
[ VERSION 2, REV. 1, 6 JUNE 72
[     MODIFIED FROM ATEXT VERSION 8/11/70
[      MODIFICATIONS BY R.H.S.   R.F.A.

EXPUNGE

TITLE           RADIK
ENTRY           RADIK,GATE1,GATEX

MACRO1          NUL; NULLL ENDM

MACRO1          IFCHANGE        (AA)
                JPSR            IFC             [CALL SUBR.
IREPEAT Q,(AA)
                SWITCHNO        (Q)
ENDI
                                0
ENDM

MACRO1          IFSWITCH        (AA)
                JPSR            IFS
IREPEAT Q,(AA)
                SWITCHNO        (Q)
ENDI
                                0
ENDM

MACRO1          SWITCHNO        (CC,BB)
DECIMAL
```

```
AA = CC
ØCTAL
                AA√K√K√K√K + BB
ENDM

MACRØ1          P (A1,A2,A3,A4,A5)
A1√B√K + A2√B + A5√B√K + A3√B√K + A4√B + 0
ENDM

MACRØ1          ØRIGIN
                P (13,106.,11,40.)
                P (10,22,37)
ENDM

ZZZ=101                         [ASCII CØDE DEFINITIØNS
IREPEAT ZZ,(A,B,C,D,E,F,G,H,I,J,K,L,M,N,Ø,Q,Q,R,S,T,U,V,W,X,Y,Z)
ZZ\. = ZZZ
ZZZ = ZZZ + 1
ENDI
P.=120

ENDCØRE = 13777                                    [ADDED CØRE FØR DISK SYSTEM
[ PIVØT DEFINITIØNS:

CLKPV = 77755;  LPNPV = 77760
EØVPV = 77756;  EØLPV = 77757
LCGAD = 77735;  LCGEA = 77736
LCGEB = 77737;  ARØPV = 77771

[EXTRA INSTRUCTIØN DEFINITIØNS:

MD10 = 30000√H;  MD11 = 31000√H;  AR05 = 25100√H
```

```
MD07 = 27000/H;   MD06 = 26000/H;   AR11 = 71100/H
MD05 = 25000/H;   ØPIØ = 00600/H;   ARBR = 62100/H
ARLS = 17500/H;   ARRS = 17440/H;   DIVI = 16440/H
MPYL = 15420/H
MPYU = 15400/H



[ MAIN ENTRY TØ PRØGRAM

GATE1:
RADIK:         JUMP           .
               MDAR'I         RADIK
               ARMD           DBUG
               JPLS           .+2
               JPSR           $AXINT
               MDAR           $LD             [SET FREE STØRAGE
               ARMD           FREE
               MDAR'L;        ENDCØRE
               ARMD           TPTR            [SETUP TEXT ALLØCATIØN PTR.
               ARMD'Ø         $ØVL1A
               ARMD'Ø         $ØVL2A
               ARMD'Ø         $ØVL3A          [UNALLØCATE AMRMX'S ØVERLAY AREAS
               MDAR'L;        TBLK            [INIT. BLØCK LISTS
               ARMD           TBLK
               MDAR'L;        GBLK
               ARMD           GBLK
               ARXØ'F
               ARMD           GBLKN           [RESET NØ. BLØCKS
               ARMD           TBLKN
               ARMD           CBLKP           [RESET CURRENT PØINTER
```

```
                ARMD           CURFG          [RESET CURSAR FLAG
                MDAR'F         CLKS           [SETUP CLOCK SUBR.
                ARMD           CLKPV
                MDAR           C1
                ARMD           MODE

GATEX:          ARX0'F
                ARMD           $GTWD1
                MD10'0'L                      [TURN ON CLOCK
                1000/H
[ START OF MAJOR FRAME LOOP

SA:             MDAR           TOLDS          [SAVE OLD SWITCHES
                ARMD           OLDS
                MDIC'0'H       C40
                S5AR'F'H                      [SAMPLE NEW SWITCHES
                MDAR'A         AMASK
                ARMD'B         NEWS
                MDIC'A'H       CM40
                S5AR'F'H
                MDAR'A         C7             [FOOT PEDALS
                MDAE'L
                MDAR           .+3            [FP1 - FS1 TEXT EDIT REQ
                ARIR'F                        [FP2 - FS8 GO EDIT
                JUMP           .+9.
                0;1;400;401;100000;100001;100400;100401
                NOOP
                MDAR'0         NEWS
                ARMD           NEWS
                ARMD           TOLDS          [SAVE FOR NEXT OLD SWITCHES
                MDX0'A         OLDS           [GET CHANGED SWITCHES
                ARMD           CNGS
```

```
        MDAR'F'N     2            [SETUP FRAME COUNTER
        ARMD         FTIME
        ARMD'0       FFLAG        [SET FRAME FLAG
        MDAR'H       $GTWD1
        JPLS         SA1          [JUMP IF COMMAND PRESENT

SB:     MDAR         DBUG
        JPLS         .+3
        MDAR'N       OKFLG
        JPAN         SE           [JUMP IF NO

        ARX0'F
        ARMD         OKFLG        [RESET FLAG
        MDAR         MODE
        MDX0'F       3            [GRAPHICS WAIT MODE
        JPLS         .+2
        JUMP         SK4          [JUMP IF YES
        MDX0'F       3'2          [TEXT WAIT MODE
        JPLS         .+2
        JUMP         SJ4          [JUMP IF YES

SE:     MDAR         MODE         [DISPATCH ON MODE
        MDAE'L
        JUMP         .+1
        ARIR'F
        JUMP         SG           [DISPLAY MODE
        JUMP         SF           [TEXT WAIT MODE
        JUMP         SF           [GRAPHICS WAIT MODE
        JUMP         SJ           [TEXT SELECT MODE
        JUMP         SK           [GRAPHICS SELECT MODE
        JUMP         SL           [TEXT EDIT MODE
```

```
         JUMP         SQ            [GRAPHICS EDIT MODE
SA1:     ARRS         3             [GET COMMAND
         MDAR'A'F     77
         MDXO'F       22            [SPECIAL COMMAND
         JPLS         .+4

         MD10'A'L
                      177777
         JUMP         $93COX        [GO DO SPECIAL COMMAND

         MDXO'F       22'20         [EDIT O.K.
         JPLS         SA3           [JUMP IF NOT

         ARMD'O       OKFLG         [SET O.K. TO EDIT FLAG

SA2:     ARXO'F
         ARMD         $GTWD1        [RESET COMMAND WORD
         JPSR         $FINSH        [SET ACCESS BIT
         JUMP         SB

SA3:     MDAR'N       MODE          [O.K. TO PROCESS COMMAND
         MDAS'F       6
         JPAN         SB            [IGNORE IF NOT
         MDAR'H       $GTWD1        [GET COMMAND
         ARRS         3
         MDAR'A'F     77
         MDXO'F       1             [IS IT VCD COMMAND
         JPLS         N2            [NO, TEST FOR JOYSTICK
         JPSR         $TRVCD        [YES, GET DIAL SETTINGS
         ARXO'F
         ARMD         CNT           [SET UP LOOP TO OBTAIN
         MDAR'F       $TVCDA        [ALL SIX VALUES
```

```
           MDAE'L;      -1
           ARMD'L;
FADD:      0
           MDAR'I'X     FADD        [GET VALUES ØF ALL
           ARRS         6           [DIALS
           NØØP
           ARMD'I       FADD
           MDAR'X       CNT
           MDXØ'L;      6
           JPLS         FADD+1
           JPSR         $RØWFW      [GET READY TØ QEND ACCRØSS
                        -0          [TØ 9300
                        $GTWD2
                        $TVCDA
                        6
           JPSR         FSWI        [TRANSMIT THE VALUES
                        7BLK
           JUMP         SB

N2:        MDXØ'F       1'2         [IS IT JØYSTK CØMMAND
           JPLS         N3          [NØ, TEST FØR F.S.
           JPSR         $TRJSB
           ARXØ'F
           ARMD         CNT         [SET UP LØØP TØ
           MDAR'F       $TJSBX      [ØBTAIN ALL VALUES
           MDAE'L;      -1
           ARMD'L;
FADD1:     0
           MDAR'X'I     FADD1
           ARRS         6
           NØØP
           ARMD'I       FADD1       [NØW GET ALL VALUES
```

```
          MDAR'X      CNT
          MDXO'L;     3
          JPLS        FADD1+1       [GET READY TO TRANSMIT
          JPSR        $ROWFW        [TO THE 9300
                      -0
                      $GTWD2
                      $TJSBX
                      3
          JPSR        FSWI          [TRANSMIT VALUES
                      7BLK
          JUMP        SB

N3:       MDXO'F      2'3           [TEST FOR F.S. COMMAND
          JPLS        N10
          MDAR        NEWS          [GET NEW QWITCHES
          ARMD        TEMP3         [STORE FOR TRANSFER
          JPSR        $ROWFW        [TO 9300
                      -0
                      $GTWD2
                      TEMP3
                      1
          JPSR        FSWI          [TRANSMIT TO 9300
                      7BLK
          JUMP        SB
N10:      MDAR'H      $GTWD1        [GET COMMAND
          ARRS        3
          MDAR'F'A    73
          MDXO'F      10            [RECIEVE TEXT BLOCK
          JPLS        SA4           [SKIP IF NOT

          MDAR        $GTWD3        [BLOCK 0
          JPLS        SA3A          [SKIP IF NOT
```

```
              MDAR'L;      ENDCORE
              ARMD         TPTR         [FREE UP BLOCK STORAGE
              MDAR'L;      TBLK
              ARMD         TBLK         [RESET TEXT BLOCKS
              ARXO'F
              ARMD         TBLKN
              MDAR         MODE         [IN TEXT WAIT OR SELECT
              MDAR'A'F     1
              JPLS         .+4          [SKIP IF NOT

              MDAR         C1
              ARMD         MODE         [SET DISPLAY MODE
              ARMD         CBLKP

SA3B:         JPSR         FSWI         [FILL SWI
                           1BLK
              JUMP         SB

SA3A:         MDAS'N       TBLKN        [BLOCK ALREADY HERE
              JPAN         SA3C         [JUMP IF YES

              MDAR         TPTR         [ENOUGH ROOM
              MDAS'F'N     26.
              MDAE'N       FREE
              JPAN         SA9          [JUMP IF NOT

              MDAR         TBLKE        [ROOM FOR ENTRY
              MDAE'N       TBLK
              MDAS'F'N     3
              JPAN         SA9          [JUMP IF NOT
```

```
        MDAR        TPTR        [ALLØCATE BLØCK
        MDAS'F'N    26.
        ARMD        TPTR
        MDAS'F'N    1
        ARMD'I'X    TBLK
        ARMD        T1
        MDAS'F      25.
        ARMD        T4
        MDAR        TBLK        [SET PØINTER TØ BLØCK
        ARMD        T2
        ARMD        T10
        MDAR        C26.        [SET SIZE
        ARMD'I'X    TBLK
        ARXØ'X      TBLKN       [INDEX NØ. TEXT BLØCKS
        ARMD'I'X    TBLK

SA3D:   JPSR        $RØWFW      [READ
                    0
                    $GTWD2
                    TBUFF
                    51.
        MDAR        $GTWD1      [GET CØUNT
        MDAS'F'N    3
        ARMD'N      T3
        MDAR        TBUFF       [GET SIZE
        ARRS        7
        MDAR'A      CMASK
        MDAE'L
        MDAR        STBL-1
        ARIR'F
        ARMD'I'H    T1          [SET SIZE
        MDAR        TBUFF       [GET INTENSITY
```

```
                MDAR'A          CMASK
                MDAE'L
                MDAR'H          ITBL-1
                ARIR'F
                MDAR'I'0'H      T1              [COMBINE
                ARMD'I'B        T1              [SAVE IN TABLE
                MDAR'H          TBUFF+1         [GET X POS.
                MDAR'0          TBUFF+2         [COMBINE WITH Y
                ARMD'I'X'B      T1              [SAVE IN LIST
                MDAR'F          TBUFF+2         [SETUP FETCH PTR.
                ARMD            T5

SA3E:           MDAR'X          T3              [ANY MORE
                JPLS            SA3F            [SKIP IF YES

                MDAR'I          T1              [INSERT EOL
                MDAR'0'H        C1
                ARMD'I          T1

SA3G:           ARX0'F
                ARMD'I'X'B      T1              [ZERO WORD
                MDAR            T1              [END OF ALLOCATION
                MDX0            T4
                JPLS            SA3G            [REPEAT IF NOT

SA3H:           MDAR'H'N        $GTWD1
                MDAR'A'F        40
                JPLS            SA3B            [GO ACKNOWLEDGE IF NOT TO EDIT

                JPSR            FSWI
                                1BLK            [RESPOND
                ARX0'F
```

```
          ARMD          NFLAG
          MDAR          C2
          ARMD          MODE          [SET TEXT WAIT MODE
          ARMD'O        OKFLG         [SET O.K. TO EDIT
          MDAR          T10           [COMPUTE BLOCK NUMBER
          MDAE'L'N,     TBLK-2
          ARLS          1
          DIVI          3
          NOOP
          ARMD          CBLKN         [SAVE
          MDAR          T10           [GET BLOCK POINTER
          JUMP          SJ5           [GO SETUP LINE

SA3F:     MDAR'I'X      T5            [FETCH WORD
          ARMD'H        T6            [SAVE
          MDAR'X        T3            [ANY MORE
          JPLS          SA3J          [JUMP IF YES

          MDAR          T6            [GET PARTIAL WORD
          MDAR'O        CB15          [SET EOL
          JUMP          SA3G+1        [GO STORE AND FINISH

SA3J:     MDAR'I'X      T5            [FETCH NEXT WORD
          MDAR'O        T6
          ARMD'I'B'X    T1            [SAVE
          MDAR          T1            [END OF ALLOCATION
          MDXO          T4
          JPLS          SA3E          [JUMP IF NOT

          MDAR'I        T1            [INSERT EOL
          MDAR'O'H      C1
          ARMD'I        T1
```

```
            JUMP        SA3H        [GO END

SA3C:       MDAR'B      $GTWD3      [GET 3*BLOCK
            MDAS        $GTWD3
            MDAE'L;     TBLK-2      [GET BLOCK POINTER
            ARMD        T2
            ARMD        T10         [SAVE BLOCK POINTER
            MDXO        CBLKP       [CURRENT BLOCK
            JPLS        .+4         [SKIP IF NOT

            MDAR        C1
            ARMD        MODE        [SET TO DISPLAY MODE
            ARMD        CBLKP       [AND CLEAR CURRENT POINTER

            MDAR'I      T2          [SET START ADDRESS
            ARMD        T1
            MDAS'F      25.         [SET END ADDR.
            ARMD        T4
            JUMP        SA3D        [GO MOVE

SA4:        MDXO'F      10'11       [TRANSMIT TEXT TO SDS
            JPLS        SA5         [SKIP IF NOT

            MDAR        $GTWD3      [GET BLOCK
            MDAS'N      TBLKN
            JPAN        .+2         [SKIP IF VALID

            JUMP        SA9         [ELSE, GIVE ERROR

            MDAR'B      $GTWD3      [GET 3 * BLOCK NO.
            MDAS        $GTWD3
            MDAE'L;     TBLK-2      [GET PTR. TO BLOCK
```

```
          ARMD          T3
          MDAR'I        T3          [GET STARTING ADDR
          ARMD          T1
          MDAR'L;       TBUFF       [SETUP ST0RE P0INTER
          ARMD          T2
          MDAR'I        T1          [GET SIZE C0DE
          JPSR          GSIZ
          ARMD'I'B      T2          [SAVE
          MDAR'I'H      T1          [GET INTENSITY
          JPSR          GINT
          MDAR'0'K      T2          [C0MBINE
          ARMD'I        T2
SA4C:     MDAR'I'X'H    T1          [FETCH W0RD
          ARRS          1
          MDAR'A        M1629
          ARMD'I'X      T2          [SAVE UPPER P0RTI0N
          MDAR'I        T1          [GET W0RD
          ARRS          1
          MDAR'A        M1629
          ARMD'I'X      T2          [SAVE L0WER P0RTI0N
          MDAR'I'H'N    T1          [E0L
          MDAR'A'F      1
          JPLS          SA4C        [REPEAT IF N0T

          MDAR'I        T1          [ANY L0WER CHARACTERS
          JPLS          SA4D        [SKIP IF YES

          MDAR          T2          [BACK UP P0INTER
          MDAS'F'N      1
          ARMD          T2

SA4D:     MDAR          T2          [C0MPUTE LENGTH
```

```
             MDAE'L'N;     TBUFF-1
             ARMD          T5            [SAVE LENGTH

SA4E:        MDAR          T2
             MDX0'F        TBUFF+50.
             JPLS          .+2           [SKIP IF NOT AT END

             JUMP          SA4F

             ARX0'F
             ARMD'I'X      T2            [ZERO WORD
             JUMP          SA4E          [REPEAT

SA4F:        MDAR          $GTWD1        [GET NO. WORDS
             MDAR'A        M1629
             ARMD          .+5           [SAVE
             JPSR          $ROWFW        [WRITE
                           -0
                           $GTWD2
                           TBUFF
                           0
             JPSR          FSWI          [FILL SWI
                           2BLK          [WITH O.K.
             JUMP          SB


SA5:         MDX0'F        11'12         [RECIEVE GRAPHICS FROM SDS
             JPLS          SA6           [SKIP IF NOT

             MDAR          $GTWD3        [BLOCK = 0
             JPLS          SA5B          [SKIP IF NOT
```

```
           MDAR         $LD
           ARMD         FREE         [UNALLOCATE GRAPHICS BLOCK STORAGE
           MDAR'L;      GBLK         [RESET GRAPHICS BLOCKS
           ARMD         GBLK
           ARX0'F
           ARMD         GBLKN
           MDAR'N       MODE         [IN SELECT OR WAIT MODE
           MDAR'A'F     1
           JPLS         .+4          [SKIP IF NOT

           MDAR         C1
           ARMD         MODE         [SET DISPLAY MODE
           ARMD         CBLKP        [RESET CURRENT POINTER

SA5A:      JPSR         FSWI         [FILL SWI
                        3BLK         [WITH O.K.
           JUMP         SB

SA5B:      MDAS'N       GBLKN        [BLOCK ALREADY HERE
           JPAN         SA5C         [JUMP IF YES

           MDAR         $GTWD1       [GET COUNT
           MDAR'A       M1629
           ARMD         SA5D+2       [SAVE FOR I/O
           MDAR         TPTR         [ENOUGH CORE
           MDAE'N       FREE
           MDAS'N       SA5D+2
           JPAN         SA9          [JUMP IF NOT

           MDAR         GBLKE        [ENOUGH FOR BLOCK HEADER
           MDAE'N       GBLK
```

```
                MDAS'F'N        3
                JPAN            SA9             [JUMP IF NØT

                MDAR            FREE            [SETUP ALLØCATIØN
                ARMD            SA5D+1
                ARMD'I'X        GBLK
                MDAR            GBLK
                ARMD            T10             [SAVE BLØCK PØINTER
                MDAR'X          GBLK            [GET PØINTER
                ARMD            T4
                MDAR            FREE
                MDAS            SA5D+2
                ARMD            FREE
                MDAS'F'N        1
                ARMD'I          T4              [SAVE END PØINTER
                ARMD'I'X        GBLK
                ARAR'X          GBLKN           [INDEX BLØCK NUMBER
SA5E:           JPSR            $RØWFW          [READ DATA
                                0
SA5D:                           $GTWD2
                                0
                                0
                MDAR'I          SA5D+1          [SETUP INTENSITY
                ARLS            3
                ARMD            T3
                ARXØ'F
                MDAS            T3
                ARRS            1
                MDAR'A          AMASK
                ARMD            T3
                MDAR'I          SA5D+1
                ARLS            3
```

```
          MDAR'A'H      C1            [GET DASH BIT
          MDAR'Ø        T3            [CØMBINE
          ARMD          T3            [STØRE IN T3
          MDAR'I'K      SA5D+1        [SET UP SCALE
          MDAR'H'A      SCMASK        [MASK
          MDAR'Ø        T3            [CØMBINE
          ARMD'I        SA5D+1        [SAVE
          MDAR'I'K'X    SA5D+1        [GET DX
          MDAR'H'A      SCMASK        [MASK
          ARMD          T5
          MDAR'I        SA5D+1        [SAVE
          ARLS          3
          MDAR'A'L;     77777         [GET DY
          MDAR'Ø        T5            [CØMBINE
          ARMD'I        SA5D+1        [SAVE
SA5F:     MDAR'I'K'X    SA5D+1        [FETCH X PART
          MDAR'A        M0010         [MASK
          ARMD          T5
          MDAR'I        SA5D+1        [FETCH Y PART
          ARLS          3
          MDAR'A'H      M0010         [MASK
          MDAR'Ø        T5            [CØMBINE
          ARMD          T5            [SAVE
          MDAR'I        SA5D+1        [GET MØVE/DRAW
          MDAR'A        C1
          MDAR'Ø        T5            [CØMBINE
          ARMD'I        SA5D+1        [RESTØRE IN BLØCK
          MDAR          SA5D+1        [END ØF BLØCK
          MDXØ'I        T4
          JPLS          SA5F          [REPEAT IF NØT

SA5G:     MDAR'I        SA5D+1        [INSERT EØL
```

```
          MDAR'Ø'H      C1
          ARMD'I        SA5D+1
          MDAR'H'N      $GTWD1
          MDAR'A'F      40            [LØAD AND EDIT
          JPLS          SA5A          [JUMP IF NØT

          JPSR          FSWI          [RESPØND
                        3BLK
          ARXØ'F
          ARMD          NFLAG         [RESET NEW BLØCK FLAG
          MDAR          C3
          ARMD          MØDE          [SET GRAPHICS WAIT MØDE
          ARMD'Ø        ØKFLG         [SET Ø.K. TØ EDIT
          MDAR          T10           [CØMPUTE BLØCK NUMBER
          MDAE'L'N;     GBLK-2
          ARLS          1
          DIVI          3
          NØØP
          ARMD          CBLKN
          MDAR          T10           [FETCH BLØCK PØINTER
          JUMP          SK5           [GØ SETUP CURRENT BLØCK

SA5C:     MDAR'B        $GTWD3        [GET 3 * BLØCK
          MDAS          $GTWD3
          MDAE'L;       GBLK-2        [GET PØINTER
          ARMD          T1
          ARMD          T10
          MDAS'F        1
          ARMD          T4
          MDAS'F        1
          ARMD          T2
          MDAR'I        T1            [GET STARTING ADDR
```

```
          ARMD      SA5D+1      [SAVE
          MDAS      $GTWD1      [ADD LENGTH
          MDAE'N'I  T2          [CUBTRACT ALL0CATI0N END
          MDAS'F'N  1
          JPAN      .+2         [JUMP IF FITS

          JUMP      SA9         [JUMP IF N0T EN0UGH R00M

          MDAR      $GTWD1      [GET LENGTH
          MDAR'A    M1629
          ARMD      SA5D+2      [SAVE
          MDAE'I    T1          [ADD 0RIGIN
          MDAS'F'N  1           [CREATE END ADDR.
          ARMD'I    T4          [SAVE
          JUMP      SA5E        [G0 READ

SA6:      MDX0'F    12'13       [TRANSMIT GRAPHICS T0 SDS
          JPLS      SA2         [IGN0RE IF N0T

          MDAR      $GTWD3      [GET BL0CK N0.
          MDAS'N    GBLKN
          JPAN      .+2         [JUMP IF 0.K.

          JUMP      SA9         [JUMP IF INVALID

          MDAR'B    $GTWD3      [GET 3 * BL0CK N0.
          MDAS      $GTWD3
          MDAE'L;   GBLK-2      [GET P0INTER
          ARMD      T1
          MDAS'F    2
          ARMD      T2
          MDAR'I    T1          [GET STARTING ADDR. - 1
```

```
          ARMD          T3
          MDAR'I        T3              [FETCH FIRST WØRD
          MDAR'A'H      C1              [EXTRACT DASH BIT
          ARRS          3
          ARMD          T5              [SAVE
          MDAR'I        T3              [GET INTEN.
          ARRS          2
          MDAR'A'L;     7777
          MDAR'Ø        T5
          ARMD          T5              [SAVE
          MDAR'I        T3              [GET SCALE
          MDAR'H'A      SCMASK          [MASK
          ARRS          6
          MDAR'Ø        T5
          ARMD          T5              [SAVE
          JPSR          $RØWFW          [WRITE FIRST WØRD
                        -0
                        $GTWD2
                        T5
                        1
          MDAR          $GTWD1          [GET NØ WØRDS
          MDAR'A        M1629
          MDAS'F'N      1
          ARMD          T5
          MDAR'X        $GTWD2          [GET BLØCK ADDR
          MDAR          C50.            [SET 50. WØRD LENGTH
          ARMD          SA6B+2

SA6A:     MDAR'F'N      49.             [SETUP WØRD CØUNT
          ARMD          T6
          MDAR'F        TBUFF-1
          ARMD          T7              [SET STØRE PØINTER
```

```
SA6C:     MDAR'I'H'X    T3        [FETCH LEFT PØRTIØN
          ARLS          3
          MDAR'A'K      M1828
          ARMD'I'X'K    T7        [SAVE
          MDAR'I        T3        [FETCH RIGHT PART
          ARRS          3
          MDAR'A        M1828
          MDAR'Ø'I      T7        [CØMBINE
          ARMD'I        T7
          MDAR'I        T3        [GET MØVE/DRAW
          MDAR'A        C1
          MDAR'Ø'I      T7        [CØMBINE
          ARMD'I        T7        [SAVE
          MDAR'X        T6        [END ØF SECTØR
          JPLS          SA6C      [REPEAT IF NØT

          MDAR'N        T5        [CØUNT LESS THAN 50
          MDAS'F        50.
          JPAN          .+3       [SKIP IF NØT

          MDAR          T5
          ARMD          SA6B+2    [SET LØWER CØUNT

          JPSR          $RØWFW    [WRITE SECTØR
                        -0
SA6B:                   $GTWD2
                        TBUFF
                        50.
          MDAR          $GTWD2
          MDAS'F        50.
          ARMD          $GTWD2
```

```
                MDAR            T5              [REDUCE REMAINING COUNT
                MDAS'F'N        50.
                ARMD            T5
                JPAN            .+2             [SKIP IF DONE

                JUMP            SA6A            [ELSE, REPEAT

                JPSR            FSWI            [FILL SWI
                                4BLK            [WITH DONE RESPONCE
                JUMP            SB



SA9:            JPSR            FSWI            [FILL SWI WITH ERROR CODE
                                5BLK
                JUMP            SB              [RETURN
[ DISPLAY MODE ENTRY

SG:             IFCHANGE (((1,SD),(2,SC)))
                JUMP            SF


SD:             MDAR            C4              [SET TEXT SELECT MODE
                ARMD            MODE
                ARMD            NFLAG           [RESET NEW FLAG
                MDAR            TBLK            [ANY BLOCKS
                MDX0'F          TBLK
                JPLS            .+2             [SKIP IF YES

                JUMP            SJ3             [ELSE, SETUP NEW BLOCK
```

```
          MDAR      C1          [SET BL0CK 1
          ARMD      CBLKN
          MDAR'L                [SETUP BL0CK 1 P0INTER
                    TBLK+1
          JUMP      SJ5

SC:       MDAR      C5          [SET GRAPHICS SELECT M0DE
          ARMD      M0DE
          ARMD      NFLAG       [RESET NEW BL0CK FLAG
          MDAR      GBLK        [BL0CK 1 PRESENT
          MDX0'F    GBLK
          JPLS      •+2         [SKIP IF YES
          JUMP      SK3         [ELSE, SET NEW BL0CK
          MDAR      C1          [SET BL0CK 1
          ARMD      CBLKN
          MDAR'L                [SET P0INTER
                    GBLK+1
          JUMP      SK5         [G0 PR0CESS

SZ:       MDAR      C1          [SET DISPLAY M0DE
          ARMD      M0DE
          ARMD      CBLKP       [RESET CURRENT P0INTER
          JUMP      SF          [G0 DISPLAY

SG1:      MDAR      C2          [TEXT EDIT REQUEST
```

```
SG3:        ARMD        MODE          [SET MODE
            JPSR        WSWI          [WRITE SWI
                        6BLK          [FOR EDIT REQUEST
            JUMP        SF            [GO DISPLAY


SG2:        MDAR        C3            [GRAPHICS EDIT REQUEST
            JUMP        SG3           [GO PROCESS
[ DISPLAY ROUTINE ENTRANCE

SF:         MDAR'F      GBLK          [SET POINTER TO GRAPHICS BLOCKS

SF1:        ARMD        BLKP
            MDXO        GBLK          [END
            JPLS        SF2           [JUMP IF NOT

            MDAR'F      TBLK          [SET START OF TEXT BLOCKS

SF4:        ARMD        BLKP
            MDXO        TBLK          [END
            JPLS        SF5           [JUMP IF NOT

            JPSR        PTRAC         [TRACK PEN
            MDAR'L                    [GET ADDR. OF MODE MESSAGE
            MDAR        TXLIN-1
            MDAS        MODE
            ARIR'F
            JPSR        DTEXT         [DISPLAY MODE MESSAGE
            MDAR        ETIME         [ERROR MESSAGE PRESENT
            JPLS        SF4A          [JUMP IF YES

SF6:        MDAR        FFLAG         [FRAME FINISHED
```

```
            JPAN        •-1         [REPEAT IF NØT

            JUMP        SA          [GØ START NEW FRAME

SF4A:       MDAR'X      ETIME       [INDEX TIMER
            MDAR'A'F    10          [TIME TØ DISPLAY
            JPLS        SF6

            MDAR'F      ERMSG       [DISPLAY MESSAGE
            JPSR        DTEXT
            JUMP        SF6         [GØ END

SF2:        MD10'L'Ø                [TURN ØN VECTØR GENERATØR
            60740√H
            MDAR'I'X    BLKP        [FETCH STARTING ADDR
            MDAR'Ø'L
            MD05
            ARMD        EØVPV       [SET VECTØR PIVØT
            MDAR'H      AMASK       [LØAD MASK
            MDAR'I'A    EØVPV       [SET SCALE
            AR11
            MD06'I      EØVPV       [SET INTENSITY
            MDAR'I'H'N  EØVPV       [FETCH DASH BIT
            MDAR'A'F    1
            JPLS        •+3         [SKIP IF NØT ØN

            MD10'Ø'L;   2000√H      [TURN ØN DASH
            MD07'I'X    EØVPV       [SET DX, DY
            MDAR        BLKP        [FETCH BLØCK PØINTER
            MDXØ        CBLKP       [CURRENT BLØCK
            JPLS        SF3         [SKIP IF NØT
```

```
          MD06'F      17777       [SET HIGH INTENSITY
          MDAR'F      PENS        [ENABLE LIGHT PEN
          ARMD        LPNPV
          MD10'Ø'L
          36√H

SF3:      ARXØ'F
          ARMD        DSDØN       [RESET DISPLAY DØNE FLAG
          MDAR'F      EØLS        [SET EØL INTERRUPT
          ARMD        EØLPV
          MDIR'X      EØVPV       [STRT DISPLAY

          MDAR'N      DSDØN       [DØNE
          JPAN        •-1

          MDAR'F      NULX        [NULL LIGHT PEN
          ARMD        LPNPV
          MD10'A'L                [TURN ØFF AVG
          01740√H
          MDAR        BLKP        [CURRENT BLØCK
          MDXØ        CBLKP
          JPLS        •+2

          JPSR        PTRAC       [TRACK PEN IF YES

          MDAR        BLKP        [UP P9INTER
          MDAS'F      2
          JUMP        SF1         [GØ TRY AGAIN

SF5:      MDAR'X      BLKP        [INDEX PØINTER
          MDXØ        CBLKP       [CURRENT BLØCK
          JPLS        SF7         [JUMP IF NØT
```

```
                MDAR'F          CRLIN-1         [DISPLAY CURRENT LINE
                JPSR            DTEXT
                JUMP            SF8

SF7:            MDAR'I          BLKP            [DISPLAY NØRMAL LINE
                JPSR            DTEXT

SF8:            MDAR            BLKP            [INCREMENT PØINTER
                MDAS'F          2
                JUMP            SF4             [GØ TRY AGAIN
[ ENTRY FØR TEXT SELECT MØDE

SJ:             IFCHANGE (((5,SJ1),(6,SJ2),(7,SJ3),(8,SG1),(4,SZ)))
                JUMP            SF


SJ4:            MDAR            C6              [SET TEXT EDIT MØDE
                ARMD            MØDE
                MDAR            CBLKN           [GET BLØCK NØ. ASCII
                JPSR            GASCI
                ARMD            TXL6A
                MDAR            CRLIN           [GET SIZE ASCII
                JPSR            GSIZ
                JPSR            GASCI
                ARMD            TXL6B           [SAVE
                MDAR            CRLIN+1         [GET X PØS ASCII
                ARRS            1
                MDAR'A          CMASK
                MDAS'F'N        15.
                JPSR            GASCI
                ARMD            TXL6C
```

```
         MDAR        CRLIN+2      [GET Y P0S ASCII
         ARRS        1
         MDAR'A      CMASK
         MDAS'F'N    22.
         ARRS        1
         JPSR        GASCI
         ARMD        TXL6D
         MDAR'B      C137         [INSERT CURSAR
         ARMD        CRLIN+3
         MDAR'F      CRLIN+3      [SET P0INTER
         ARMD        CP0S
         MDAR'F'N    7            [SETUP CURSAR C0UNTER
         ARMD        CURCT
         MDAR'F'N    4
         ARMD        REPCT        [SET REPEAT C0UNTER
         MDAR'N      NFLAG        [NEW BL0CK
         JPAN        SJ4A         [JUMP IF N0T

         MDAR        TBLK         [UP ALL0CATI0N P0INTER
         MDAS'F      3
         ARMD        TBLK
         MDAR        TPTR
         MDAS'F'N    26.
         ARMD        TPTR
         ARAR'X      TBLKN

SJ4A:    MDAR'L                   [SETUP F0REGR0UND/BACKGR0UND
         JUMP        SF
         ARMD        $WT1
         ARX0'F
         ARMD        CFLAG        [RESET CHARACTER FLAG
SJ4B:    JPSR        $ICHTY       [INPUT CHARACTER
```

```
                                   2
              MDBR                $TTYC           [FETCH CHARACTER
              MDBR'A              CMASK
              BRMD                CHAR            [SAVE CHARACTER
              ARMD'0              CFLAG           [SET CHARACTER FLAG
              JUMP                SJ4B            [G0 GET AN0THER

SJ1:          MDAR                TBLKN           [ANY M0RE BL0CKS
              MDX0                CBLKN
              JPLS                .+2             [SKIP IF YES

              JUMP                SF

              MDAR                NFLAG           [NEW BL0CK
              JPAN                SF              [SKIP IF YES

              ARAR'X              CBLKN           [INDEX BL0CK NUMBER
              MDAR                CBLKP           [UP P0INTER
              MDAS'F              3

SJ5:          ARMD                CBLKP
              MDAR'I              CBLKP           [FETCH ADDR. 0F BL0CK
              ARMD                T1
              MDAR'F              CRLIN+3         [SETUP LINE P0INTER
              ARMD                T2
              MDAR'I              T1              [FETCH FIRST W0RD
              ARMD                CRLIN           [SAVE INT. AND SIZE
              MDAR'I'X'H          T1              [FETCH X P0S
              MDAR'A              M1528           [MASK
              ARMD                CRLIN+1
              MDAR'I              T1              [SETUP Y P0S.
```

```
              MDAR'A         M1528
              ARMD           CRLIN+2
SJ9:          MDAR'I'H       T1              [CHECK FOR EOL
              MDAR'A'F       1
              JPLS           SJ9A

              MDAR'I'X'K     T1              [GET FIRST CHARACTER
              ARLS           2
              MDAR'A         M2228
              ARMD'I'X       T2              [SAVE
              MDAR'I'H       T1              [GET SECOND CHAR.
              MDAR'A         M2228
              ARMD'I'X       T2              [SAVE
              MDAR'I'K       T1              [GET THIRD CHAR.
              ARLS           2
              MDAR'A'H       M2228
              ARMD'I'X'H     T2              [SAVE
              MDAR'I         T1              [GET FOURTH CHAR.
              MDAR'A         M2228
              ARMD'I'X       T2              [SAVE
              JUMP           SJ9             [GO REPEAT

SJ9B:         ARX0'F                         [STORE NULLS
              ARMD'I'X       T2

SJ9A:         MDAR           T2              [END OF LIST
              MDX0'F         CRLIN+100.
              JPLS           SJ9B            [REPAAT IF NOT

SJ5A:         MDAR'H         C1              [STORE EOL
              ARMD           CRLIN+100.
              ARX0'F
```

```
            ARMD        CRLIN+3     [INSERT CURSAR
            MDAR        CBLKN       [GET ASCII F0R BL0CK NUMBER
            JPSR        GASCI
            ARMD        TXL4A
            JUMP        SF

SJ2:        MDAR        CBLKP       [BL0CK 1
            MDX0'F      TBLK+1
            JPLS        .+2         [SKIP IF N0T
            JUMP        SF

            ARMD'H      NFLAG       [RESET NEW BL0CK FLAG
            MDAR        CBLKN       [DECREMENT CURRENT BL0CK
            MDAS'F'N    1
            ARMD        CBLKN
            MDAR        CBLKP       [DECREMENT BL0CK P0INTER
            MDAS'F'N    3
            JUMP        SJ5         [G0 PR0CESS

SJ3:        MDAR        TPTR        [R00M
            MDAE'N      FREE
            MDAS'F!N    27.
            JPAN        SJ6         [JUMP IF N0T

            MDAR        TBLKE       [R00M F0R ENTRY
            MDAE'N      TBLK
            MDAS'F'N    3
            JPAN        SJ6         [JUMP IF N0T

            MDAR        TBLK        [SETUP ALL0CATI0N
            MDAS'F      1
```

```
ARMD          T1
ARMD          CBLKP
MDAR          TPTR
MDAS'F'N      27.           [SETUP ADDRESS
ARMD'I        T1
ARMD'0        NFLAG         [SET NEW BLOCK FLAG
MDAR          TBLKN
MDAS'F        1             [GET NEXT BLOCK NO.
ARMD          CBLKN         [SAVE
MDAR          C26.
ARMD'I'X      T1            [SET LENGTH
ARX0'F
ARMD'I'X      T1
MDAR'L                      [SETUP CLEAR COMMAND
ARMD          CRLIN
ARMD          .+2           [SAVE

ARX0'F
NOOP                        [RESET WORD
MDAR'X        .-1           [INDEX
MDX0'F        CRLIN+100.
JPLS          .-4           [REPEAT IF NOT END

MDAR'L                      [SETUP X POS
P (0,0,11,20)
ARMD          CRLIN+1
MDAR'L                      [SETUP Y POS
P (0,0,13,24.)
ARMD          CRLIN+2
MDAR'L                      [SETUP SIZE
P (0,0,0,22)
ARMD          CRLIN
```

```
                JUMP            SJ5A            [GO FINISH

SJ6:SK6: MDAR'F'N               200.            [SET ERROR MESSAGE TIMER
                ARMD'L
ETIME:                          0
                JUMP            SF              [GO DISPLAY
[ GRAPHICS SELECT MODE PROCESSOR

SK:             IFCHANGE (((5,SK1),(6,SK2),(7,SK3),(8,SG2),(4,SZ)))
                JUMP            SF              [DISPLAY IF NO CHANGE

SK4:            MDAR            C7              [SET GRAPHICS EDIT MODE
                ARMD            MODE
                MDAR            COFF            [SET CURSAR  OFF
                ARMD            TXL7B
                MDAR            CBLKN           [GET BLOCK NO. TEXT
                JPSR            GASCI
                ARMD            TXL7A
                ARXO'F
                ARMD            TXL7C           [RESET FULL
                ARMD            CURFG           [RESET CURSAR ON FLAG
                ARMD            SVFLG           [RESET SAVED VECTOR FLAG
                ARMD            TEPFG           [AND TRACK ENDPOINT MODE FLAG
                MDAR            CBLKP           [SET POINTERS TO ENTRY
                MDAR'A          AMASK
                MDAS'F          1
                ARMD            CREND
                MDAS'F          1
                ARMD            ABEND
                MDAR'I          CREND           [FULL
                MDXO'I          ABEND
```

```
                JPLS          .+3           [SKIP IF N0T

                MDAR          CFULL         [SET  FULL  IN MESSAGE
                ARMD          TXL7C

                MDAR'N        NFLAG         [NEW BL0CK
                JPAN          SF            [JUMP IF N0T

                MDAR          GBLK          [FREEZE ALL0CATI0N
                MDAS'F        3
                ARMD          GBLK
                MDAR'I        CBLKP         [SET CURRENT END
                MDAS'F        2
                ARMD'I        CREND
                MDAS'F'N      2
                ARMD          CPTR
                MDAR          TEMP2
                ARMD'I        CPTR
                MDAR          TEMP1
                ARMD'I'X      CPTR          [DX,DY = 0
                MDAR'H        C1            [SET E0L IN LIST
                ARMD'I'X      CPTR
                MDAR          TPTR          [SET MAX END
                ARMD'I        ABEND
                ARAR'X        GBLKN         [INDEX BL0CK N0.
                JUMP          SF            [G0 DISPLAY

SK1:            MDAR          GBLKN         [ANY M0RE BL0CKS
                MDX0          CBLKN
                JPLS          .+2           [SKIP IF YES
                JUMP          SF            [ELSE, DISPLAY
```

```
        MDAR        NFLAG       [NEW BLØCK
        JPAN        SF          [DISPLAY IF YES

        ARAR'X      CBLKN       [INDEX CURRENT BLØCK NØ.
        MDAR        CBLKP       [INCREMENT PØINTER
        MDAS'F      3

SK5:    ARMD        CBLKP       [SAVE PØINTER
        MDAR        CBLKN       [GET ASCII FØR BLØCK NØ.
        JPSR        GASCI
        ARMD        TXL5A       [SAVE IN MESSAGE
        JUMP        SF          [GØ DISPLAY

SK2:    MDAR        CBLKN       [BLØCK 1 CURRENT
        MDXØ'F      1
        JPLS        .+2         [SKIP IF NØT

        JUMP        SF          [ELSE, GØ DISPLAY

        ARMD'H      NFLAG       [RESET NEW BLØCK FLAG
        MDAR        CBLKN       [DECREMENT BLØCK NØ.
        MDAS'F'N    1
        ARMD        CBLKN
        MDAR        CBLKP       [DECREMENT BLØCK PØINTER
        MDAS'F'N    3
        JUMP        SK5         [GØ SET NEW BLØCK

SK3:    MDAR        TPTR        [FETCH CØRE LIMIT
        MDAE'N      FREE
        MDAS'F'N    10.         [SEE IF 10 WØRDS AVAILABLE
        JPAN        SK6         [JUMP IF NØT
```

```
                MDAR            GBLKE           [R00M F0R ENTRY
                MDAE'N          GBLK
                MDAS'F'N        3
                JPAN            SK6             [JUMP IF N0T

                MDAR            GBLK            [ALL0CATE BL0CK (TEMP0RARY)
                MDAS'F          1
                ARMD            T1
                ARMD            CBLKP
                MDAR            FREE            [SET START 0F BL0CK
                ARMD'I          T1
                ARMD'0          NFLAG           [SET NEW BL0CK FLAG
                MDAR            GBLKN           [SET CURRENT BL0CK N0
                MDAS'F          1
                ARMD            CBLKN
                JUMP            SK5+1           [G0 PR0CESS
[TEXT EDIT M0DE PR0CESS0R

SL:             MDAR'N          CFLAG           [CHARACTER PRESENT
                JPAN            SM              [JUMP IF N0T

                ARMD            CFLAG           [RESET FLAG
                MDAR            CHAR            [CHARACTER = RUB0UT
                MDX0'F          177
                JPLS            SL1             [JUMP IF N0T

                MDAR            CP0S            [CURSAR AT BEG. 0F LINE
                MDX0'F          CRLIN+3
                JPLS            .+2             [SKIP IF N0T

                JUMP            SM              [SKIP M0VE IF YES
```

```
          JPSR          PSHDN          [MØVE BUFFER DØWN
          MDAR          CPØS           [BACK UP CURSAR
          MDAS'F'N      1
          ARMD          CPØS
          MDAR          C40            [INSERT CURSAR CHARACTER
          ARMD'I'B      CPØS
          JUMP          SM             [GØ CHECK SWITCHES

SL1:      MDXØ'F        177'5          [CHAR = CNTL  E
          JPLS          SL2            [JUMP IF NØT

          JPSR          PSHDN          [MØVE CHARACTERS DØWN
          MDAR          C40            [RE-INSERT CURSAR
          ARMD'I'B      CPØS
          JUMP          SM             [GØ GET SWITCHES

SL2:      MDXØ'F        5'6            [CHAR = CNTL  F
          JPLS          SL3            [JUMP IF NØT

          MDAR          CPØS           [CURSAR AT END ØF LINE
          MDAS'F        1
          ARMD          T1
          MDXØ'F        CRLIN+100.
          LSAR'I        T1             [ØR CHARACTER AFTER = NULL
          JPLS          .+2            [SKIP IF Ø.K.

          JUMP          SM             [ELSE, READ SWITCHES

          MDAR'I        T1             [SWAP CHARACTERS
          ARMD'I        CPØS
          MDAR          C40            [INSERT CURSAR
          ARMD'I'X'B    CPØS
```

```
          JUMP        SM          [GO CHECK SWITCHES

SL3:      MDXO'F      6'22        [CHAR = CNTL  R
          JPLS        SL35        [SKIP IF NOT

          MDAR        CPOS        [CURSAR AT BEG. OF LINE
          MDAS'F'N    1
          ARMD        T1
          MDXO'F      CRLIN+2
          JPLS        .+2         [SKIP IF NOT

          JUMP        SM

          MDAR'I      T1          [SWAP CHARACTERS
          ARMD'I      CPOS
          MDAR        T1          [MOVE CURSAR FORWARD
          ARMD        CPOS
          MDAR'B      C40
          ARMD'I      CPOS        [INSERT CURSAR
          JUMP        SM

SL35:     MDXO'F      22'15       [CHAR= CAR. RET.
          JPLS        SL4         [SKIP IF NOT
          JUMP        SU          [ACT LIKE AN END EDIT

SL4:      MDAR        CHAR        [CHECK VALID DISPLAY CHAR.
          MDAR'A'F    140
          LSXO        .-1
          JPLS        .+2         [SKIP IF O.K.

          JUMP        SM          [ELSE, IGNORE
```

```
            MDAR          CRLIN+99.      [BUFFER FULL
            JPLS          SM             [SKIP IF YES

            MDAR'F        CRLIN+99.      [SETUP MØVE PØINTERS
            ARMD          T1
            MDAR'F        CRLIN+98.
            ARMD          T2

SL5:        MDAR'I        T2
            ARMD'I        T1             [MØVE CHARACTER UP
            MDAR          T2
            MDAS'F'N      1              [DECREMENT PØINTERS
            ARMD          T2
            MDAR          T1
            MDAS'F'N      1
            ARMD          T1
            MDXØ          CPØS           [END ØF MØVE
            JPLS          SL5            [JUMP IF NØ

            MDAR          CHAR           [INSERT CHARACTER
            ARMD'I'B      CPØS
            MDAR          C40            [INSERT CURSAR
            ARMD'I'X'B    CPØS
            MDAR          CRLIN+99.      [BUFFER FULL
            JPLS          .+2            [SKIP IF YES

            JUMP          SM

            MDAR          CFULL
            ARMD          TXL6E          [INSERT  FULL  IN MESSAGE

SM:         IFSWITCH (((12,SM1)))
```

```
            JUMP          SN             [JUMP IF 12. NOT ON

SM1:        MDAR'X        REPCT          [INDEX REPEAT COUNTER
            JPLS          SN             [JUMP IF NOT END OF COUNT

            MDAR'F'N      4              [SET COUNTER
            ARMD          REPCT
            ARXO'F
            ARMD          TOLDS          [RESET SWITCH SAVE

SN:
            IFCHANGE (((4,SU),(9,SN1),(10,SN2),(13,SN3)))
            IFCHANGE (((14,SN4),(15,SN5),(16,SN6)))
            JUMP          SP             [JUMP IF NO CHANGE

SN1:        MDAR          CRLIN          [CURRENT SIZE = 3
            JPSR          GSIZ
            MDXO'F        3
            JPLS          .+2            [SKIP IF NOT

            JUMP          SP

            MDXO'F        3              [RESTORE
            MDAE'L;       STBL

SN1A:       ARMD          T1             [SET POINTER TO SIZE
            MDAR          CRLIN          [MASK OUT OLD
            MDAR'A'H      AMASK
            MDAR'O'I'B    T1             [INSERT NEW
            ARMD          CRLIN          [SET BACK IN LIST
            JPSR          GSIZ           [GET NEW NUMBER
            JPSR          GASCI          [GET ASCII STRING
```

```
                ARMD            TXL6B           [SET IN MESSAGE
                JUMP            SP

SN2:            MDAR            CRLIN           [GET SIZE
                JPSR            GSIZ
                MDXO'F          1               [SIZE 1
                JPLS            .+2             [JUMP IF NOT

                JUMP            SP

                MDXO'F          1               [RESTORE
                MDAE'L;         STBL-2
                JUMP            SN1A            [GO SET NEW SIZE

SN3:            MDAR'F          SXT             [SET POINTER FOR X INCREASE

SN3A:           ARMD            T1
                MDAR            CRLIN+1         [GET X POS
                MDAR'A'B        CMASK
                MDXO'I          T1              [AT END
                JPLS            .+2             [SKIP IF NOT

                JUMP            SP

                MDAR            CRLIN+1         [MOVE POINTER
                MDAS'I'H        T1
                ARMD            CRLIN+1
                ARRS            1
                MDAR'A          CMASK           [MASK POSITION
                MDAS'F'N        15.             [OFFSET
                JPSR            GASCI           [GET ASCII FOR MESSAGE
                ARMD            TXL6C
```

```
        JUMP        SP

SN4:    MDAR'F      SXT+1       [SET FOR NEGATIVE X MOVE
        JUMP        SN3A

SXT:    2√H         (96.+15.√B)
        77775√H     (16.√B)

SN5:    MDAR'F      SYT         [SET FOR POSITIVE Y MOVE

SN5A:   ARMD        T1
        MDAR        CRLIN+2     [GET POSITION
        MDAR'A'B    CMASK
        MDX0!I      T1          [AT END
        JPLS        .+2         [SKIP IF NOT

        JUMP        SP

        MDAR        CRLIN+2     [ADJUST
        MDAS'I'H    T1
        ARMD        CRLIN+2
        ARRS        1
        MDAR'A      CMASK
        MDAS'F'N    22.
        ARRS        1
        JPSR        GASCI       [GET ASCII FOR MESSAGE
        ARMD        TXL6D
        JUMP        SP

SN6:    MDAR'F      SYT+1       [SET FOR NEGATIVE Y MOVE
        JUMP        SN5A
```

```
SYT:          4√H            (22•+80•√B)
              77773√H        (24•√B)

SP:           MDAR'X         CURCT          [INDEX CURSAR BLINK COUNTER
              JPLS           SF             [GO DISPLAY IF NOT ZERO

              MDAR'F'N       7
              ARMD           CURCT          [RESET COUNTER
              MDAR'I         CPOS           [COMPLEMENT CHARACTER
              MDXO'L;        (40'137√B)
              ARMD'I         CPOS
              JUMP           SF             [GO DISPLAY


SU:           MDAR'I         CBLKP          [SETUP POINTER TO BLOCK
              ARMD           T1
              MDAR'L;        CRLIN+3        [SET POINTER TO LINE
              ARMD           T2
              MDAR'L                        [SETUP SWI WORD 1
              00210√H        2
              ARMD           SU9A
              MDAR           CRLIN          [SAVE INT• AND SIZE
              ARMD'I         T1
              MDAR'H         CRLIN+1        [SAVE X AND Y POS
              MDAR'O         CRLIN+2
              ARMD'I'X       T1

SU1:          MDAR           T2             [END OF LIST
              MDAE'N'L;      CRLIN+98•
              JPAN           SU2            [JUMP IF NOT

SU3:          MDAR'I         T1             [INSERT EOL
```

```
        MDAR'0'H    C1
        ARMD'I      T1
        MDAR        CRLIN       [GET SIZE
        JPSR        GSIZ
        ARLS        10.         [SHIFT
        N00P
        MDAR'0      CBLKN       [0R IN BL0CK NUMBER
        ARMD        SU9A+1      [SAVE F0R TRANSMIT
        MDAR        CRLIN+1     [GET X P0SITI0N
        MDAR'A'B    CMASK
        ARMD'K      SU9A+2
        MDAR        CRLIN+2     [GET Y P0SITI0N
        MDAR'A'B    CMASK
        MDAR'0'K    SU9A+2      [C0MBINE
        ARMD        SU9A+2      [SAVE F0R TRANSMIT
        JPSR        WSWI        [WRITE SWI BUFFER
                    SU9A
        MDAR        C1          [SET DISPLAY M0DE
        ARMD        M0DE
        ARMD        CBLKP       [RESET CURRENT P0INTER
        JUMP        SF          [G0 DISPLAY
SU2:    JPSR        SUS         [GET W0RD
        JPLS        .+2         [SKIP IF 0.K.
        JUMP        SU3         [ELSE, G0 END
        ARAR'X      SU9A        [INDEX W0RD C0UNTER
        ARMD'B      T3          [SAVE W0RD
        ARAR'X      T2
        JPSR        SUS         [GET NEXT CHAR
        MDAR'0'K    T3
```

```
            ARAR'B'X      T2         [INDEX AND SHIFT
            ARMD'B        T3
            JPSR          SUS        [GET NEXT CHARACTER
            JPLS          .+2        [SKIP IF NOT ZERO

            JUMP          .+2

            ARAR'X        SU9A       [INDEX COUNT IF NON-ZERO

            MDAR'O'K      T3         [COMBINE
            ARMD'B        T3
            ARAR'X        T2         [INDEX
            JPSR          SUS        [GET NEXT CHAR.
            MDAR'O'K      T3         [COMBINE
            ARMD'I'X      T1         [SAVE IN BUFFER
            ARAR'X        T2         [INDEX POINTER FOR NEXT
            JUMP          SU1        [GO TRY AGAIN


SUS:        JUMP          .          [GET WORD SUBR.
            MDAR          T2         [AT CURSAR POSITION
            MDXO          CPOS
            JPLS          .+2        [SKIP IF NOT

            ARAR'X        T2         [INDEX POINTER IF YES

            MDAR'I        T2         [FETCH CHARACTER
            MDAR'A'B      CMASK      [MASK
            MDIR          SUS        [RETURN

SU9A:                     0
```

```
                    0
                    0


[ GRAPHICS EDIT MODE PROCESSOR

SQ:         JPSR        PTRAC           [TRACK CURSAR IF ON
            IFCHANGE (((4,SV),(9,SQ1),(10,SQ2),(11,SQ3)))
            IFCHANGE (((13,SQ4),(14,SQ5),(15,SQ6),(12,SQ9)))
            IFSWITCH (((1,LTRAN),(2,RTRAN),(5,UTRAN),(6,DTRAN)))
            IFSWITCH (((3,BIG),(7,SMALL)))
            IFSWITCH (((8,ROTAT)))
            JUMP        SR

SQ1:        MDAR'I      CREND           [SET LAST COORD AS TRACKING COORD.
            ARMD        T1
            MDAR'I      T1
            ARMD        TRCRD
            ARMD'O      CURFG           [SET CURSAR ON FLAG
            MDAR        CON
            ARMD        TXL7B
            JUMP        SR

SQ2:        ARXO'F
            ARMD        CURFG           [RESET CURSAR
            MDAR        COFF
            ARMD        TXL7B
            JUMP        SR

SQ3:        MDAR'N      SVFLG           [VECTOR SELECTED
            JPAN        SR              [JUMP IF NOT
```

```
          MDAR      SVPTR     [SAVE SELECTED VECTOR ADDR.
          ARMD      TEPTR
          JPSR      RSV       [RESET SELECTED VECTOR
          ARMD'0    TEPFG     [SET TRACK ENDPOINT MODE
          ARMD'0    CURFG     [AND CURSAR ON
          MDAR      CON
          ARMD      TXL7B
          MDAR'I    TEPTR     [FETCH VECTOR TO TRACK
          ARMD      TRCRD     [SET FOR CURSAR
          MDAR'A    C1H1
          ARMD      TEPMB     [SAVE MODE BITS
          JUMP      SR

SQ6:      MDAR      TRCRD     [SET SKETCH COORDINATES
          ARMD      SKCRD
          JUMP      SR

SQ7:      NOOP
          JUMP      SR

SQ4:      ARX0'F              [SET DRAW BIT

SQ4D:     ARMD      T3
          MDAR'N    CURFG     [CURSAR ON
          JPAN      SS        [JUMP IF NOT

SQ4C:     MDAR'I    CREND     [FULL
          MDX0'I    ABEND
          JPLS      SQ4B      [JUMP IF NOT

SQ4A:     JPSR      SRNK      [SHRINK TABLE
```

```
          MDAR'I       CREND        [STILL FULL
          MDX0'I       ABEND
          JPLS         .+2          [JUMP IF N0

          JUMP         SS

SQ4B:     ARX0'F
          ARMD         TXL7C        [RESET  FULL
          MDAR         TRCRD        [FETCH TRACK C00RD.
          MDAR'A       CM1H1        [RESET FLAG BITS
          MDAR'0'H     C1           [SET E0L
          MDAR'0       T3           [SET M0VE/DRAW
          ARMD         T2           [SAVE
          MDAR'I       CREND        [FETCH END 0F LIST ADDR.
          ARMD         T1
          MDAR'I'N     T1           [FETCH W0RD
          MDAR'0'H     C1           [REM0VE E0L
          ARMD'I'N     T1           [REST0RE
          MDAR         T2           [SAVE NEW VECT0R
          ARMD'I'X     T1
          MDAR         T1           [SET NEW END ADDR
          ARMD'I       CREND
          MDX0'I       ABEND        [FULL N0W
          JPLS         SS           [JUMP IF N0T

          MDAR         CFULL
          ARMD         TXL7C        [SET IN MESSAGE
          JUMP         SS

SQ5:      MDAR         C1           [SET DRAW BIT
          JUMP         SQ4D
```

```
SQ9:        MDAR'I       CBLKP        [FETCH BLOCK ADDR
            ARMD         T1
            MDAR'I       T1           [COMPLEMENT DASH
            MDXO'H       C1
            ARMD'I       T1
            JUMP         SR

LTRAN:      MDAR         CBLKP        [LEFT TRANSLATE ROUTINE
            ARMD         TEMP         [GET FWA AND SAVE
            MDAR'I       TEMP
            ARMD         TEMP
            MDAR         AMASK        [MASK AND BRING UP
            MDAR'I'A'X   TEMP         [DX FOR MOD
            ARMD         TEMPT
            MDAR'I       TEMP
            MDAE         XLINC        [MODIFY DX
            MDAR'A'H     AMASK
            MDAR'O       TEMPT        [COMBINE WITH DY
            ARMD'I       TEMP         [PUT BACK IN BLOCK
            ARMD         TEMP1        [ALSO FOR STND ALONE
            JUMP         SF

RTRAN:      MDAR         CBLKP        [RT TRANSLATE ROUTINE
            ARMD         TEMP         [GET FWA AND SAVE
            MDAR'I       TEMP
            ARMD         TEMP
            MDAR         AMASK        [MASK AND BRING UP
            MDAR'I'A'X   TEMP         [DX FOR MOD
            ARMD         TEMPT
            MDAR'I       TEMP
            MDAE         XRINC        [MODIFY DX
            MDAR'A'H     AMASK
```

```
            MDAR'O       TEMPT        [COMBINE WITH DY
            ARMD'I       TEMP         [PUT BACK IN BLOCK
            ARMD         TEMP1        [ALSO FOR STND ALONE
            JUMP         SF

UTRAN:      MDAR         CBLKP        [UP TRANS ROUTINE
            ARMD         TEMP         [GET FWA AND SAVE
            MDAR'I       TEMP
            ARMD         TEMP
            MDAR'I'X     TEMP
            MDAE         UPINC        [MODIFY DY
            ARMD'I       TEMP         [PUT BACK IN BLOCK
            ARMD         TEMP1        [ALSO FOR STND ALONE
            JUMP         SF

DTRAN:      MDAR         CBLKP        [DOWN TRANS ROUTINE
            ARMD         TEMP         [GET FWA AND SAVE
            MDAR'I       TEMP
            ARMD         TEMP
            MDAR'I'X     TEMP
            MDAE         DNINC        [MODIFY DY
            ARMD'I       TEMP         [PUT BACK IN BLOCK
            ARMD         TEMP1        [ALSO FOR STND ALONE
            JUMP         SF

BIG:        MDAR         CBLKP        [MAGNIFY
            ARMD         TEMPS        [GET FWA AND SAVE
            MDAR'I       TEMPS
            ARMD         TEMPS
            MDAR'I       TEMPS
            MDAE         PSCALE       [MODIFY SCALE
            ARMD'I       TEMPS        [PUT BACK IN BLOCK
```

```
            ARMD       TEMP2      [ALSØ FØR STND ALØNE
            JUMP       SF

SMALL:      MDAR       CBLKP      [DIMINISH SIZE
            ARMD       TEMPS      [GET FWA AND SAVE
            MDAR'I     TEMPS
            ARMD       TEMPS
            MDAR'I     TEMPS
            MDAE       MSCALE     [MØDIFY SCALE
            ARMD'I     TEMPS      [PUT BACK IN BLØCK
            ARMD       TEMP2      [ALSØ FØR STND ALØNE
            JUMP       SF

RØTAT:      MDAR       ANG        [RØTATIØN RØUTINE
            JPSR       $SNCØS     [PASS ARG TØ SNCØS
            MDAR       CBLKP      [GET FWA ØF BLØCK
            ARMD       TEMPR
            MDAR'I     TEMPR
            ARMD       TEMPR
            MDAR'I'X   TEMPR      [INDEX PAST FIRST,
            MDAR'I'X   TEMPR      [SECØND WØRDS
RØTA:       ARAR'H                [NØW BRING UP EACH
            MDAR'A     C1         [DATA WØRD AND APPLY
            JPLS       SF         [RØTATIØN ØF CØØRD
            MDAR'I     TEMPR      [EQUATIØNS AS PER
            MDAR'A     EMASK      [CRC HANDBØØK
            ARMD       BITS
            MDAR'I     TEMPR
            MDAR'A'L;  77777JH    [SAVE M/D, AND EØL
            ARMD       XTEMP      [BITS
            MDAR'I     TEMPR
            MDAR'A'L;  77777
```

```
            ARMD            YTEMP
            MDAR            $CØSN
            MPYU            XTEMP
            NØØP
            ARMD            XCØS
            MDAR            $SINE
            MPYL            YTEMP
            NØØP
            ARMD            YSIN
            MDAR            $CØSN
            MPYL            YTEMP
            NØØP
            ARMD            YCØS
            MDAR            $SINE
            MPYU            XTEMP
            NØØP
            ARMD            XSIN
            MDAE            YCØS
            ARAR'H'F
            MDAR'A'L;       77777
            ARMD            XSYC
            MDAR'N          YSIN
            MDAE            XCØS
            MDAR'A'L;       77777JH
            MDAS            XSYC
            MDAR'A          RMASK
            MDAR'Ø          BITS
            ARMD'I          TEMPR
            MDAR'I'X        TEMPR
            JUMP            RØTA
SR:         IFSWITCH (((11,SR1),(15,SR2)))
            JUMP            SS
```

```
SR1:        MDAR'N      TEPFG       [TRACK ENDPOINT MODE
            JPAN        ST          [JUMP IF NOT

            MDAR        TRCRD       [FETCH CURRENT TRACK COORDS.
            MDAR'A      CM1H1       [REMOVE FLAG BITS
            MDAR'O      TEPMB       [INSERT CORRECT ONES
            ARMD'I      TEPTR       [SET INTO VECTOR
            JUMP        ST

SR2:        ARXO'F
            MDAS'H      TRCRD       [COMPUTE DELTA Y
            MDAS'H'N    SKCRD
            ANXO        CM0
            MDAS'N      LIMIT
            JPAN        .+2         [SKIP IF NOT EXCEEDED

            JUMP        SR2A        [ELSE, GO INSERT VECTOR

            ARXO'F
            MDAS        TRCRD       [COMPUTE DELTA X
            MDAS'N      SKCRD
            ANXO        CM0
            MDAS'N      LIMIT
            JPAN        ST          [JUMP IF NOT EXCEEDED

SR2A:       MDAR        C1          [SETUP FOR INSERTING DRAW
            ARMD        T3
            MDAR        TRCRD       [SET NEW SKETCH COORD.
            ARMD        SKCRD
            JUMP        SQ4C        [GO INSERT VECTOR
```

```
SS:        IFSWITCH  (((11,ST)))
           ARXØ'F                      [RESET TRACK ENDPØINT MØDE
           ARMD        TEPFG
           JUMP        ST

SV:        JPSR        SRNK            [SHRINK TABLE
           MDAR        CBLKP           [FETCH ENTRY ADDR.
           MDAS'F      1               [GET END ADDR
           ARMD        T2
           MDAR'N      NFLAG           [NEW BLØCK
           JPAN        SV1             [JUMP IF NØT

           MDAR        T2              [SETUP ENTRY PØINTER
           MDAS'F      1
           ARMD        T3
           MDAR'I      T2              [SET ALLØCATIØN END
           ARMD'I      T3
           MDAS'F      1
           MDAR'A      AMASK
           ARMD        FREE            [SET FREE STØRAGE

SV1;       MDAR'I      T2
           MDAE'N'I    CBLKP           [CØMPUTE SIZE
           MDAS'F      2
           MDAR'Ø'L                    [SETUP CØMMAND WØRD
           00220√H
           ARMD        SV9A            [SAVE FØR TRANSMIT
           MDAR        CBLKN           [GET BLØCK NØ.
           ARMD        SV9A+1          [SAVE
           JPSR        WSWI            [WRITE SWI BUFFER
                       SV9A
```

```
                MDAR        C1          [SET DISPLAY MODE
                ARMD        MODE
                ARMD        CBLKP       [RESET CURRENT POINTER
                ARMD        CURFG       [RESET CURSAR ON

ST:             JPSR        PTRAC       [TRACK PEN
                JUMP        SF          [GO DISPLAY

SV9A:                       0
                            0
                            0


[ PEN TRACKING SUBROUTINE

TI = 300;
MTI = -TI 77777
SI = 40
MSI = -SI 77777

PTRAC:          JUMP        .
                JPSR        PTCK
                MDIR        PTRAC
PTCK:           JUMP        .           [ENTRY
                MDAR'N      CURFG       [CURSAR ON
                ANIR        PTRAC       [RETURN IF NOT

                MDAR'F      NUL         [NULL LIST PIVOT
                ARMD        EOLPV
                MD10'O'L                [TURN ON AVG
                60776√H
```

```
         ARXO'F
         ARMD           LPN2
         ARXO'F                        [CLEAR AR OVF PIVOT
         ARMD           AROPV
         MDAR'F         LPN2           [SETUP PEN IN ERRUPT
         ARMD           LPNPV
         MD06'F         0
         MDAR'F         PTPT-1         [SETUP OFFSET TABLE PTR.
         ARMD           PTRCP
         MDAR           TRCRD          [FETCH TRACT COORD
         ARMD           TRCRP          [SAVE FOR UPDATE
         MDAR'A         CM1            [MASK OUT DRAW
         MDAR'O'H       C1             [SET EOL
         JPSR           LOD5
         0; 0; 0; 0; 0;
         JPSR           LOD5
         MDAR           TRCRD          [SET FOR DRAWING DOT
         MDAR'O         C1H1
         NOOP
         NOOP
         JPSR           LOD5
         MDAR           TRCRD          [FETCH COORD
         MDAE'L                        [MOVE TO UPPER RIGHT
         400/H          400
         MDAR'A         CM1            [REMOVE DRAW
         MDAR'O'H       C1             [SET EOL
         NOOP
         NOOP
         JPSR           LOD5
IREPEAT Q,(7740000401,7740077401,40077401,40000401)
         MDAR           TRCRD          [FETCH COORDS.
         MDAE'L                        [OFFSET
```

```
             Q
             MDAR'O        C1H1          [SET DRAW AND EOL
             NOOP
             NOOP
             NOOP
             ARAR'X        PTRCP
             JPSR          LOD5
     ENDI
             NOOP
             NOOP
             NOOP
             NOOP
             MDAR          LPN2
             JPLS          PTCK9
             MDAR'F        LPN22
             ARMD          LPNPV
             MDAR          TRCRD
             MDAE'L
             1750JH        500
             MDAR'A        CM1
             MDAR'O'H      C1
             JPSR          LOD5
     NOCARRET
     IREPEAT Q,(115201523,2033,7662401523,7603000501,
             7603077301,7662476255,75745,115276255,
             175077301,175000501);
     CARRET
             NOOP
             NOOP
             NOOP
             NOOP
             MDAR          TRCRD
```

```
                MDAE'L
                Q
                MDAR'O          C1H1
                ARAR'X          PTRCP
                JPSR            LOD5
ENDI
                NOOP
                NOOP
                NOOP
                NOOP
PTCK9:          MDAR            TRCRP
                ARMD            TRCRD
                MDAR'F          NULX
                MD10'A'L                        [RESET AVG
                1000√H
                ARMD            LPNPV           [NULL LIGHT PEN PIVOT
                MDIR            PTCK            [RETURN
                                0
PTPT:           SI;MSI√H;MSI;SI√H

                145101010;57601777;7671201777;7632701012
                7603077700;7632776776;7671276000;57676000
                145076770;175077700
                NOOP; NOOP

[ INTERRUPT SUBR. FOR LIGHT PEN

LPN2:                           0
                ARMD            LPN2A           [SAVE (AR)
                MDAR            TRCRP           [ADJUST POSITION
                MDAE'I          PTRCP
                ARMD            TRCRP           [SAVE
```

```
                MDAR            LPN2A           [RESTØRE (AR)
                JUMP'I          LPN2            [RETURN AND UNLØCK

LPN22:                          0
                ARMD            LPN2A
                MDAR            TRCRP
                MDAE'I          PTRCP
                ARMD            TRCRP
                MDAR'F          NULX
                ARMD            LPNPV
                MDAR            LPN2A
                JUMP'I          LPN22
LPN2A:                          0
PTRCP:                          0
TRCRP:                          0

LØD5:           JUMP            .
                ARMD'L;         0
                MD05            .-1
                MDIR            LØD5
[ CØMMUNICATIØNS SUBRØUTINES..........

[WRITE SWI BUFFER SUBR.

WSWI:           JUMP            .               [ENTRY
                MDAR            DBUG
                JPLS            WSWIX
                MDAR'I          WSWI            [FETCH AGT ADDRESS
                ARMD            .+4             [SAVE
                JPSR            $RØWFW          [WRITE
                                -0
                                $SWIA
```

```
                                    0
                                    3
               OPIO                 43005          [GIVE INTERRUPT
WSWIX:
               MDIR'X               WSWI           [RETURN

[ FILL SWI BUFFER SUBR.

FSWI:          JUMP                 .              [ENTRY
               MDAR                 DBUG
               JPLS                 FSWIX
               ARXO'F                              [RESET COMMAND FLAG
               ARMD                 $GTWD1
               JPSR                 $FINSH         [SET ACCESS BIT
               MDAR'I               FSWI           [FETCH ADG.
               ARMD                 .+2            [SAVE FOR WSWI CALL
               JPSR                 WSWI           [WRITE SWI BUFFER
                                    0
FSWIX:
               MDIR'X               FSWI           [RETURN

[SWI BLOCKS TO BE TRANSMITTED:

1BLK:          100√H
2BLK:          110√H
3BLK:          120√H
4BLK:          130√H
5BLK:          140√H
6BLK:          200√H
7BLK:          10√H
```

```
[ TEXT MODE MESSAGES.........

[DISPLAY MODE

TXL1:           1√H


[TEXT WAIT MODE

TXL2:
                ORIGIN
                P (T.,E.,X.,T.)
                P (40,E.,D.,I.)
                P (T.,40,R.,E.)
                P (Q.,U.,E.,S.)
                P (T.,,,,1)


[GRAPHICS WAIT MODE

TXL3:
                ORIGIN
                P (G.,R.,A.,P.)
                P (H.,I.,C.,S.)
                P (40,E.,D.,I.)
                P (T.,40,R.,E.)
                P (Q.,U.,E.,S.)
                P (T.,,,,1)
```

```
[TEXT SELECT MØDE

TXL4:
                 ØRIGIN
                 P (T.,E.,X.,T.)
                 P (40,B.,L.,Ø.)
                 P (C.,K.,40,S.)
                 P (E.,L.,E.,C.)
                 P (T.,40,M.,Ø.)
                 P (D.,E.,40,40)
                 P (40,40,40,40)
                 P (40,40,40,40)
                 P (40,40,B.,L.)
                 P (Ø.,C.,K.,40)
TXL4A:           0
                 1√H


[GRAPHICS SELECT MØDE

TXL5:
                 ØRIGIN
                 P (G.,R.,A.,P.)
                 P (H.,I.,C.,S.)
                 P (40,B.,L.,Ø.)
                 P (C.,K.,40,S.)
                 P (E.,L.,E.,C.)
                 P (T.,40,M.,Ø.)
                 P (D.,E.,40,40)
                 P (40,40,40,40)
                 P (40,40,B.,L.)
                 P (Ø.,C.,K.,40)
```

```
TXL5A:          0
                1√H

[TEXT EDIT MODE

TXL6:
                ORIGIN
                P (T.,E.,X.,T.)
                P (40,E.,D.,I.)
                P (T.,40,M.,O.)
                P (D.,E.,40,40)
                P (40,40,B.,L.)
                P (O.,C.,K.,40)
TXL6A:          0
                P (40,40,40,S.)
                P (I.,Z.,E.,40)
TXL6B:          0
                P (40,40,40,X.)
                P (40)
TXL6C:          0
                P (40,40,40,Y.)
                P (40)
TXL6D:          0
                P (40,40,40,40)
TXL6E:          0
                1√H

[GRAPHICS EDIT MODE

TXL7:
```

```
                ØRIGIN
                P (G.,R.,A.,P.)
                P (H.,I.,C.,S.)
                P (40,E.,D.,I.)
                P (T.,40,M.,Ø.)
                P (D.,E.,40,40)
                P (40,40,B.,L.)
                P (Ø.,C.,K.,40)
TXL7A:          O
                P (40,40,40,40)
                P (C.,U.,R.,S.)
                P (A.,R.,40)
TXL7B:          P (Ø.,F.,F.)
                P (40,40,40,40)
TXL7C:          O
                1√H

ERMSG:          P (13,106.,11,140)
                P (22,37,S.,T.)
                P (Ø.,R.,A.,G.)
                P (E.,40,F.,U.)
                P (L.,L.,,,1)
[GET ASCII CHARACTERS FØR VALUE SUBR.

GASCI:          JUMP            .
                MDAR'A          CMASK           [MASK VALUE
                ARLS            1
                DIVI            10.             [DIVIDE BY 10
                NØØP
                ARMD'H          GAS1            [SAVE REMAINDER
                JPLS            .+2             [SKIP IF NØT ZERØ
```

```
                MDAR'N          C20             [SET -20 FØR SPACE

                MDAS'F          60              [CREATE ASCII CHARACTER
                MDAR'A          CMASK
                ARMD'K          GAS2            [SAVE
                ARXØ'F
                MDAS            GAS1            [FETCH REMAINDER
                ARRS            1
                MDAS'F          60              [CREATE CHARACTER
                MDAR'A          CMASK
                MDAR'Ø'B        GAS2            [CØMBINE FIRST CHARACTER
                ARAR'B'F
                MDIR            GASCI           [RETURN

GAS1:                           0
GAS2:                           0


[GET SIZE NUMBER SUBRØUTINE

GSIZ:           JUMP            .
                JPLS            .+3             [SKIP IF NØT MIDDLE SIZE

                MDAR            C2              [SET SIZE = 2
                MDIR            GSIZ

                MDXØ'F          (22√B)          [SMALL
                JPLS            .+3             [SKIP IF NØT

                MDAR            C1              [SET SIZE = 1
                MDIR            GSIZ
```

```
        MDAR        C3          [SET SIZE = 3
        MDIR        GSIZ

[GET INTENSITY NUMBER SUBR.

GINT:   JUMP        .
        JPLS        .+3         [SKIP IF NOT MIDDLE

        MDAR        C2          [SET INT. = 2
        MDIR        GINT

        MDXO'F      (36/B)      [DIM
        JPLS        .+3         [SKIP IF NOT

        MDAR        C1          [SET INT. = 1
        MDIR        GINT

        MDAR        C3          [SET INT. = 3
        MDIR        GINT

[ RESET SELECTED VECTOR SUBR.

RSV:    JUMP        .
        MDAR'N      SVFLG       [VECTOR SELECTED
        ANIR        RSV         [RETURN IF NOT

        ARMD        SVFLG       [RESET FLAG
        MDAR'I      SVPTR       [FETCH VECTOR
        MDAR'O      C1          [SET DRAW BIT ON
```

```
                ARMD'I          SVPTR           [RESTØRE
                MDAR'F          DUMMY
                ARMD            SVPTR           [SETUP DUMMY PØINTER
                MDIR            RSV             [RETURN

[ PEN INTERRUPT SUBRØUTINE

PENS:           JUMP            .
                ARMD            ARSAV           [SAVE (AR)
                MDAR            CURFG           [CURSAR ØN
                JPAN            PENS2           [RETURN IF YES

                JPSR            RSV             [RESET ANY SELECTED VECTØR
                MDAR            EØVPV           [FETCH VECTØR PIVØT
                MDAS'F'N        2               [BACK UP
                ARMD            SVPTR           [SAVE

PENS1:          MDAR'I'X'N      SVPTR           [FETCH VECTØR
                MDAR'A'F        1               [DRAW
                JPLS            .-2             [REPEAT IF NØT

                MDAR'I          SVPTR           [FETCH VECTØR
                MDXØ            C1              [MAKE IT A MØVE
                ARMD'I          SVPTR           [RESTØRE
                ARMD'Ø          SVFLG           [SET SELECTED VECTØR FLAG
PENS2:          MDAR            ARSAV           [RESTØRE (AR)
                JUMP'I          PENS            [RETURN AND UNLØCK

ARSAV:                          0
```

```
[PUSHDOWN CHARACTER LIST SUBROUTINE

PSHDN:          JUMP            .
                MDAR            CPOS            [SETUP POINTERS
                ARMD            T1
                MDAS'F'N        1
                ARMD            T2
                MDXO'F          CRLIN+98.       [END
                JPLS            .+2             [SKIP IF NOT

                JUMP            PSH2

PSH1:           MDAR'I'X        T1              [MOVE CHARACTER DOWN
                ARMD'I'X        T2
                MDAR            T1              [END OF MOVE
                MDXO'F          CRLIN+99.
                JPLS            PSH1            [REPEAT IF NOT

PSH2:           ARXO'F
                ARMD            CRLIN+99.       [SET LAST CHAR = NULL
                ARMD            TXL6E           [REMOVE  FULL
                MDIR            PSHDN           [RETURN


[SHRINK VECTOR LIST SUBROUTINE

SRNK:           JUMP            .
                JPSR            RSV             [RESET ANY SELECTED VECTORS
                MDAR'I          CBLKP           [GET START OF BLOCK
                ARMD            SRN9            [SET POINTERS
                ARMD            SRN0
                ARXO'F                          [RESET MOVE FLAG
```

```
            ARMD          SRN8
            MDAR'X        SRN0          [INDEX TØ GET 2ND
            MDAR'X        SRN9          [WØRDN

SRN1:       MDAR'I'X      SRN0
            MDAR'A'F      1             [DRAW
            JPLS          SRN2          [JUMP IF YES

            MDAR          SRN8          [SARED MØVE
            JPLS          .+2           [SKIP IF YES

            JUMP          .+3           [IGNØRE IF NØT


            ARXØ'F
            ARMD          TXL7C         [REMØVE  FULL
            MDAR          SRN0          [SAVE MØVE PØINTER
            ARMD          SRN8

SRN3:       MDAR          SRN0          [END ØF LIST
            MDXØ'I        CREND
            JPLS          SRN1          [JUMP IF NØT

            JPSR          SRNS          [CØPY ANY MØVE
            MDAR          SRN9          [SET NEW CURRENT END
            ARMD'I        CREND
            MDIR          SRNK          [RETURN

SRN2:       JPSR          SRNS          [CØPY ANY MØVE
            MDAR'I        SRN0          [CØPY THIS DRAW
            ARMD'I'X      SRN9
            JUMP          SRN3          [GØ CHECK END
```

```
SRN0:                   0
SRN9:                   0
SRN8:                   0

SRNS:       JUMP        .
            MDAR        SRN8        [ANY MOVE SAVED
            JPLS        .+2         [SKIP IF YES

            MDIR        SRNS        [ELSE, RETURN

            MDAR'I      SRN8        [FETCH MOVE
            ARMD'I'X    SRN9        [SAVE
            ARXO'F                  [RESET MOVE FLAG
            ARMD        SRN8
            MDIR        SRNS        [RETURN

[ DISPLAY TEXT LINE SUBROUTINE

DTEXT:      JUMP        .
            MDAS'F'N    1
            ARMD        LCGAD       [SAVE LCG PIVOT
            MDIC'A      CM14        [RESET LCG
            MDAR'F      LCGS        [SETUP END LIST PIVOTS
            ARMD        LCGEA
            ARMD        LCGEB
            MD07        ZERO        [RESET DX, DY
            MD10'O'L                [TURN ON 2D AND SCOPES
            40760√H
            MD05        ZERO        [MOVE BEAM TO CENTER
            ARXO'F
```

```
                ARMD            T1              [RESET END FLAG
                MDIC'0          C10             [TURN ON LCG
                MD10'A'L                        [RESET AVG
                -40000√H
                MDAR'X          T1              [FETCH END FLAG
                ANIR            DTEXT           [RETURN IF DONE
                JPLS            .-2             [REPEAT IF NOT TIME-OUT

                MDIC'A          CM14            [RESET LCG
                MDIR            DTEXT           [RETURN

[ LCG INTERRUPT SUBROUTINE

LCGS:           JUMP            .
                MDIC'A          CM14            [RESET LCG
                ARMD'0          T1              [SET END FLAG
                JUMP'I          LCGS            [RETURN AND UNLOCK

[ FRAME CLOCK INTERRUPT SUBROUTINE

CLKS:           JUMP            .
                ARMD            CLKAR           [SAVE (AR)
                MDAR'H'X        FTIME           [INDEX TIMER
                JPAN            .+2             [SKIP IF NOT THREE TICKS

                ARMD            FFLAG           [SET FRAME FLAG IF DONE

                MDAR            CLKAR           [RESTORE (AR)
                JUMP'I          CLKS            [RETURN AND UNLOCK
```

```
CLKAR:                          0

[ AVG END ØF LIST INTERRUPT SUBRØUTINE

EØLS:           JUMP            .
                ARMD'Ø          DSDØN           [SET DØNE FLAG
                JUMP'I          EØLS            [RETURN AND UNLØCK
[ CHECK SWITCH CHANGE SUBRØUTINE


IFC:            JUMP            .               [ENTRY
                MDAR            CNGS            [FETCH CHANGED SWITCHES

IFC2:           ARMD            IFC1            [SAVE
                MDAR'I          IFC             [FETCH ARG.
                ARAR'X          IFC
                JPLS            .+2             [SKIP IF ARGUMENT

                MDIR            IFC             [ELSE, RETURN

                ARMD            IFC9            [SAVE
                ARAR'F'K                        [GET NØ.
                MDAR'A          VMASK
                MDAE'L                          [CREATE SHIFT
                ARLS            13.
                ARMD            .+3             [SAVE
                MDAR'L'N                        [FETCH SWITCHES
IFC1:                           0
                NØØP
                NØØP
                NØØP
```

```
            JPAN        IFC2+1      [REPEAT IF NØT ØN

            JUMP'I      IFC9        [ELSE, DISPATCH

IFC9:                   0



[ IF SWITCH ØN SUBRØUTINE

IFS:        JUMP        .           [ENTRY
            MDAR        IFS         [SAVE ENTRY
            ARMD        IFC
            MDAR        NEWS        [FETCH SWITCHES ØN
            JUMP        IFC2        [GØ PRØCESS



[ NULL SUBRØUTINES

NUL:        JUMP        .
            JUMP'I      NUL

NULX:       JUMP        .
            JUMP'I      NULX
[ TABLES

TBUFF:      LØC (.+51.)             [CØMM. BUFFER

            P (0,0,0,10)
```

```
CRLIN:       LØC (.+100.)                 [CURRENT TEXT LINE BUFFER
             P (0,0,0,0,1)

STBL:                      22             [SIZE TABLE
                           0
                           23

ITBL:                      36             [INTENSITY TABLE
                           0
                           37

TBLK:                      .              [TEXT BLØCK LIST
             LØC           (.+244.)
TBLKE:                     .

GBLK:                      .              [GRAPHICS BLØCK LIST
             LØC           (.+154.)
GBLKE:                     .

TXLIN:                     TXL1           [MØDE MESSAGE TABLE
                           TXL2
                           TXL3
                           TXL4
                           TXL5
                           TXL6
                           TXL7


[ CØNSTANTS
ANG:                       1000
RMASK:                     7777677776
EMASK:                     0000100001
```

```
XRINC:                          00070√H
XLINC:                          77707√H
DNINC:                          77707
UPINC:                          00070
PSCALE:                         00070√H
MSCALE:                         77706√H
SCMASK:                         77760
AMASK:                          77777
CMASK:                          177
VMASK:                          77
ZERO:                           0
C1:                             1
C2:                             2
C3:                             3
C4:                             4
C5:                             5
C6:                             6
C7:                             7
C10:                            10
C20:                            20
C40:                            40
C137:                           137
C26.:                           26.
C50.:                           50.
CB15:                           40000
C1H1:           1√H             1
CM0:                            -0
CM1:                            -1
CM1H1:          77776√H         77776
CM14:                           -14
CM40:                           -40
CFULL:          P (F.,U.,L.,L.)
COFF:           P (O.,F.,F.)
```

```
CØN:           P (Ø.,N.)
M1629:                        37777
M0010:         77760√H
M1528:                        77776
M1828:                        7776
M2228:                        376
LIMIT:                        100
[ VARIABLES
TEMPT:                        0
TEMP3:                        0
CNT:                          0
TEMP:                         0
TEMP1:                        0
TEMPS:                        0
TEMP2:                        3777610000
BITS:                         0
XTEMP:                        0
YTEMP:                        0
XCØS:                         0
YSIN:                         0
XSIN:                         0
YCØS:                         0
XSYC:                         0
TEMPR:                        0

T1:0;T2:0;T3:0;T4:0;T5:0;T6:0;T7:0;T10:0
TPTR:                         0
CPTR:                         0
SVPTR:                        0
FREE:                         0
TBLKN:                        0
GBLKN:                        0
```

```
CBLKN:                        0
CBLKP:                        0
MODE:                         0
TOLDS:                        0
OLDS:                         0
NEWS:                         0
CNGS:                         0
FTIME:                        0
FFLAG:                        0
OKFLG:                        0
NFLAG:                        0
DSDON:                        0
BLKP:                         0
CPOS:                         0
CURCT:                        0
REPCT:                        0
CFLAG:                        0
CHAR:                         0
CURFG:                        0
SVFLG:                        0
TEPFG:                        0
CREND:                        0
ABEND:                        0
TRCRD:                        0
TEPTR:                        0
TEPMB:                        0
SKCRD:                        0
DUMMY:                        0
DBUG:                         0
IFPASS1  TYPEOUT '
LENGTH = '  TYPEOCT (.);    ENDC
TERMINATE
```

LIST OF REFERENCES

1. Licklider, J.C., "Graphic Input- A Survey of Techniques," in Computer Graphics, Gruenberger, F., ed., Thompson Book Company, 1967.

2. Johnson, C.I., "Principles of Interactive Systems," IBM Systems Journal, V. 7, pp. 147-174, 1968.

3. Machover, C., "Computer Graphics in the United States," in Computer Graphics, Parslow, R., and others, ed., Plenum Press, 1969.

4. Licklider, J.C., "Man-Computer Symbiosis," IRE, Transactions on Human Factors in Electronics, pp. 4-11, March, 1960.

5. Sutherland, I.E., "Sketchpad A Man-Machine Graphical Communication System," AFIPS Conference Proceedings 1963 Spring Joint Computer Conference, V. 23, Spartan Books Incorporated, 1963.

6. Johnson, T.E., "Sketchpad III A Computer Program For Drawing in Three Dimensions," AFIPS Conference Proceedings 1963 Spring Joint Computer Conference, V. 23, pp. 347-355, Spartan Books Incorporated, 1963.

7. Jacks, E.L., "A Laboratory for the Study of Graphical Man-Machine Communications," AFIPS Conference Proceedings 1964 Fall Joint Computer Conference, V. 26, Spartan Books Incorporated, 1964.

8. Brown, G.D., and Bush, C.H., The Integrated Graphics System for the IBM 2250, The Rand Corporation, 1968.

9. Prince, D.M., Interactive Graphics for Computer Aided Design, pp. 41-42, Addison-Wesley, 1971.

10. Sherr, S., Fundamentals of Display System Design, pp. 1-25, Wiley and Sons, 1969.

11. Bartee, J.C., Digital Computer Fundamentals, 2nd ed., p. 11, McGraw-Hill, 1966.

12. Miller, R.B., "Response Time in Man-Computer Conversational Transactions," AFIPS Conference Proceedings 1968 Fall Joint Computer Conference, V. 23, The Thompson Book Company, 1968.

13. Smith, L.B., "The Use of Interactive Graphics to Solve Numerical Problems," Communications of the ACM, V. 13, pp. 625-634, 1970.

14. Coons, S.A., "An Outline of the Requirements for a Computer-Aided Design System," AFIPS Conference Proceedings 1963 Spring Joint Computer Conference, V. 23, pp. 299-303, Spartan Books Incorporated, 1963.

15. The Rand Corporation Memorandum Rm-5825-pr, Pogo: Programmer-Oriented Graphics Operation, B.W. Boehm and others, pp. 1-34, June 1969.

16. Naval Ship Research and Development Center Technical Note AL-168, Computer Interactive Graphics Systems, G.W. Ellis, pp. 13-16, June 1970.

17. Williams, R., "A Survey of Data Structures for Computer Graphics Systems," Computing Survey, V. 3, pp. 1-17, March 1971.

18. McKeeman, W.M., and others, A Compiler Generator, pp. 217-219, Prentice Hall, 1970.

19. Knuth, D.E., Fundamental Algorithms V. 1, pp. 229-303, Addison-Wesley, 1968.

20. Baskin, H.B., and Morse, S.P., "A Multilevel Modeling Structure for Interactive Graphic Design," IBM Systems Journal, V. 7, pp. 218-228, 1968.

21. Electrical Engineering Computer Laboratory, Manual for the Fortran User, pp. 1-1, 1-11, R.D. Delaura, 10 February 1972.

22. Van Dam, A., and Bergeon, R., "Software Capabilities of the Adage Graphics Terminals," in Advanced Computer Graphics, Parslow, R., and others, ed., Plenum Press, 1971.

23. Department of Electrical Engineering Computer Laboratory TN-13-1-A, XDS/AGT Interface Description, W.A. Thomas, pp. 1-4, 9 February 1972.

24. Electrical Engineering Computer Laboratory Program Report PR-3-2-A, TRJSB, pp. 1-2, W.A. Thomas, 16 February 1972.

25. Electrical Engineering Computer Laboratory Program Report PR-3-3-A, TRVCD, pp. 1-2, W.A. Thomas, 16 February 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Dr. George A. Rahe, Code 52Ra Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	LCDR Ralph H. Stowell, Jr., USN 8201 Ector Court Annandale, Virginia 22003	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

RADIK: An Interactive Graphic and Text Editor System

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*

Master's Thesis; June 1972

5. AUTHOR(S) *(First name, middle initial, last name)*

Ralph Henry Stowell, Jr.

6. REPORT DATE June 1972	7a. TOTAL NO. OF PAGES 154	7b. NO. OF REFS 25
8a. CONTRACT OR GRANT NO. b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

This thesis contains a survey of the hardware requirements and the desirable characteristics of an interactive Computer Graphics System. There is a detailed evaluation of the resident system in the Computer Laboratory and an attempt is made through programming, to provide a software capability which allows for full utilization of all hardware in the Computer Graphics System. This program, known as RADIK, is a highly interactive graphics and text editor developed for utilization on a computer system composed of an XDS 9300 and two Adage AGT-10 graphics terminals. In addition, this thesis includes a discussion on the development of the system as well as complete operating instructions.

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Interactive Graphics						
Interactive Programs						
Computer Graphics						

DD FORM 1 NOV 65 1473 (BACK)

S/N 0101-807-6821

Security Classification

A-31409